Cutting plane selection for quadratic semidefinite outer-approximations with a data driven approach and advanced analytics

National Technical University of Athens

Asimina Marousi, 05114003

14 July 2020

Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται την επίλυση προβλημάτων βελτιστοποίησης τετραγωνικού προγραμματισμού, που αποτελεί το πιο στοιχειώδες παράδειγμα μη γραμμικού προγραμματισμού. Οι σύγχρονοι αλγόριθμοι βελτιστοποίησης βασίζονται σε μεθόδους αποικοδόμησης και χαλάρωσης. Εισάγωντας *επίπεδα αποκοπής*, τα οποία λειτουργούν σαν κατώτερα εφικτά όρια, προσεγγίσουν εξωτερικά το αρχικό πρόβλημα. Η επιλογή των κατάλληλων *επιπέδων αποκοπής* είναι γνωστό ως *πρόβλημα διαχωρισμού*. Στόχος της παρούσας εργασίας είναι η βελτίωση των αλγόριθμων αναφοράς, αξιολογώντας, σε πραγματικό χρόνο, την συμπληρωματικότητα των επιλεγμένων *επιπέδων αποκοπής*, επιλύνωντας δηλαδή το *πρόβλημα διαχωρισμού*. Προκειμένου να αναδειχθούν κοινές ιδιότητες των *επιπέδων αποκοπής*, γίνεται χρήση των εργαλείων της μηχανικής μάθησης και πιο συγκεριμένα, τεχνικές ομαδοποίησης. Κάθε *επίπεδο αποκοπής* αποτελεί μια ανισότητα και προκύπτει από έναν συνδιασμό των μεταβλητών του προβλήματος βελτιστοποίησης. Τα διαφορετικά σημεία από τα οποία προκύπτουν *επίπεδα αποκοπής* αποτελούν τον πληθυσμό των μεθόδων ομαδοποίησης. Οι τεχνικές που χρησιμοποιούνται είναι οι αλγόριθμοι k-means και Agglomerative clustering, χρησιμοποιώντας την Ευκλείδια απόσταση για τον υπολογισμό της απόστασης. Επιπλέον εισάγεται μια νέα απόσταση η ονομαζόμενη *νόρμα της συγγένειας* η οποία χρησιμοποιείται τόσο αυτούσια όσο και σε συνδιασμό με τον αλγόριθμο k-means δημιουργώντας έναν υβριδικό αλγόριθμο ομαδοποίησης. Τα υπολογιστικά πειράματα πραγματοποιήθηκαν για ένα εύρος προβλημάτων, διφορετικές διαστάσεις του αρχικού προβλήματος (40-100 μεταβλητές) και

διαφορετική πυκνότητας της μήτρας του τετραγονικού προγράμματος (25% -75%). Τα αποτελέσματα αναδεικνύουν πως η χρήση τεχνικών ομαδοποίησης επιφέρει σημαντικές βελτιώσεις σε σχέση με τους αλγόριθμους αναφοράς.

Abstract

Quadratic programming (QP) is a base case study of non linear optimization problems. State-of-the-art(SoA) algorithms have proposed decomposition and relaxations to outer-approximate the QP by creating extra inequality constraints known as cutting planes. The selection of the cutting planes is known as the *separation problem*. The aim of the study has been to accelerate the performance of the approach against the SoA by evaluating the complementarity of the selection of cutting planes, subsequently exploring online improvements in the *separation problem*. An approach is proposed to commit cut selection with data analytics thus connecting the method with emerging technologies in data models and machine learning. Each cutting plane is an inequality and is structured by a combination of the variables of the original optimization problem. The different points creating the cutting planes are the population of the clustering algorithms. Clustering methods are deployed using k-means and Agglomerative clustering, as well as clustering based on the tailor-made affinity norm and Hybrid algorithms combining k-means and affinity norm. The results reveal significant improvements against SoA for a wide range of problems that vary in size (40-100 dimensions) and sparsity (25-75% density).

Acknowledgements

I would like to thank my professor Antonis Kokossis for his guidance and giving the opportunity to pursue this research. I would like to express my gratitude to Dr. Ruth Misener for giving me the background and motivation of this study and professor Nilay Shah for giving me the opportunity to work in his group in Imperial College London.

I would like to thank my family for their support in every step of this journey. Thodori, thank you for keep pushing me forward and always being there for me.

Contents

Chapter 1

Introduction

1.1 Mathematical Optimization

Mathematical optimization is the process of finding the input variables of a function, within an allowed domain, in order to reach the minimum or maximum value of this function. Optimization problems vary in the type of objective function and domains of the variables. The general formulation for an optimization problem(P) is the following:

$$P: \begin{aligned} & \min_x f(x) \\ & s.t.\ h(x) = 0 \\ & g(x) \leq 0 \\ & x \in \mathbb{R}^n \end{aligned}$$

Finding the optimum can be secured at some occasions where the constraints are creating a convex domain for the variables, which enables the continuous transition from one incumbent solution to the next. Nevertheless, in a great number of applications the domain is not convex

and the optimization becomes much more complex.

The optimization problems are categorized based the formulation of the objective function and the domain resulted by the constraints. In this section we are presenting the basic categories of optimization problems and their basic characteristics. In the following section we are introducing the key concept of convex analysis and providing the founding theorems. In order address the optimization problem we are outlining the basic numerical and iterative optimization methods. However, in cases where the detection of global optimum is desired, numerical and iterative optimization methods are not always suitable. In section 1.4, we are justifying why global optimization is a challenge and in the proceeding section a different perspective, data driven, is proposed to exploit the data generated in global optimization algorithms. Finally we are introducing the motivation of this research followed by the structure of the diploma thesis.

1.1.1 Convex programming

Concex programming studies optimization problems with convex(minimization) or concave (maximization) the case objective function and the constraint domain is convex. Least squares, linear programming, semidefinite programming and some cases of quadratic programming are general convex optimization problems. Interior-point methods are commonly used to address convex problems since there is no general analytical algorithm for the solution of convex optimization problems. According to Boyd and Vandenberghe in[5] solving convex optimization is not a mature technology for all cases of problems, e.g. nonlinear convex problems. The difficulty lies in the recognition of a convex function or a nonconvex function that can be transformed into convex.

- **Least-squares problems**[5] A least squares problem is an optimization problem with

no constraints and an objective which is a sum of squares of terms of the form $\alpha_i^T x - b_i$:

$$\min_x \ f(x) = \|Ax - b\|_2^2 = \sum_{i=1}^{k} (\alpha_i^T x - b_i)^2$$

where $A \in \mathbb{R}^{k \times n}$ (with $k \geq n$), α_i^T are the rows of A, and the vector $x \in \mathbb{R}^n$ is the optimization variable. The solution of least-squares problem can be reduced to analytically solving a set of linear equations.

$$(A^T A)x = A^T b$$

For least-squares problems there are good algorithms and software implementations for solving the problem to high accuracy, with very high reliability [5]. Least-squares problems are the formulation of regression analysis, optimal control and many parameter estimation and data fitting methods such as clustering techniques. To assess if an optimization problem is a least-squares problem, we only need to verify that the objective is a quadratic function and test if the quadratic form is positive semidefinite.

- **Linear programming** (LP) is a type of convex programming and addresses problems with linear objective function and linearly formulated constraints. A general linear program has the form :

$$\begin{aligned} \min_x \ & c^T x + d \\ s.t. \ & Gx \leq h \\ & Ax = b \end{aligned}$$

where, $G \in \mathbb{R}^{m \times n}$ and $A \in \mathbb{R}^{p \times n}$

For LP there is not a characteristic analytical formula but there is a variety of methods such as Simplex method and interior point methods[5]. In many cases the original

optimization problem needs to be reformulated to attain the final LP structure. LP is commonly examined in operations research for a variety of problems such as planning, routing, scheduling, assignment, and design.

- **Semidefinite programming** (SDP) is a subfield of convex optimization where the underlying variables are semidefinite matrices. It is a generalization of linear and convex quadratic programming. In contrast with LP where the variables of objective function are real, SDP is formulated using real-valued vectors. The inequality constraints of LP are replaced by semidefineteness constraints on matrix variables of SDP. A general SDP problem can be formulated as :

$$\min_{x^1,\cdots,x^n \in \mathbb{R}^n} \sum_{i=1,\ j=1}^{n} c_{i,j}(x^i \cdot x^j)$$
$$s.t. \sum_{i=1,\ j=1}^{n} \alpha_{i,j,k}(x^i \cdot x^j) \leq b_k,\ \forall k \in 1, \cdots, m$$

where $c_{i,j}, \alpha_{i,j,k}, b_k$ are real numbers.

The above equation can be reformulated as

$$\min_{X \in S^n} \langle C, X \rangle_{\mathbb{S}^n}$$
$$s.t.\ \langle A_k, X \rangle_{\mathbb{S}^n} \leq b_k\ \forall k \in 1, \cdots, m$$
$$X \succeq 0$$

where $\mathbb{S}^n$ is the space of all $n \times n$ real symmetric matrices, C is given by $\frac{c_{i,j}+c_{j,i}}{2}$, A_k is a symmetric $n \times n$ matrix where $A_{i,j,k} = \frac{a_{i,j,k}+a_{j,i,k}}{2}$ and X is positive semidefinite.

In general a symmetric $n \times n$ real matrix M is positive semidefinite if and only if the scalar $z^T M z \geq 0, \quad \forall z \in \mathbb{R}^n - \{0\}$

1.1.2 Nonlinear programming

(NLP) studies the general case in which the objective function or the constraints or both composed of nonlinear elements. This may or may not be a convex program. Whether the program is convex, or not, affects the difficulty of solving it. Geometrically, nonlinear programs can significantly differ from LP. The optimal solution may occur:

- at an interior point of the feasible region
- on the boundary of the feasible region,which is not an extreme point, or
- at an extreme point of the feasible region

Simplex method is based on searching extreme points thus is not suitable for NLP problems. Pivoting and other algebraic procedures are commonly used by NLP algorithms to replace the original problem by an approximating linear one, these nonlinear algorithms renew the approximations of each iteration based on the solution of the last one [6]. Depending on the nature of NLP objective function and constraint there is an abundance of different optimization algorithms and solvers.

1.1.3 Mixed integer linear programming

(MILP) studies linear programs in which some variables are constrained to take only integer values. This is not convex and, in general, much more difficult than regular linear programming. In many application of MILP the integer variables are binary (0-1 variables). The MILP

formulation with 0-1 variables is stated as

$$\min_{x,y} c^T x + d^T y$$

$$s.t.\ Ax + By \leq b$$

$$x \geq 0,\ x \in X \subseteq \mathbb{R}^n$$

$$y \in \{0,1\}^q$$

where: x is a vector of n continuous variables

y is a vector of q 0-1 variables

c, d are correspondingly $(n \times 1), (q \times 1)$ vectors of parameters

A, B are matrices of the appropriate dimension

b is a vector of p inequalities.

The MILP problem of the above formulation belongs to the class of NP-complete problems [11]. The difficulty arises due to the combinatorial nature of the domain of y variables. For every combination of 0-1 y allocation arises a different optimization problem that is to be solved for optimality. Even if we would attempt to solve all the problems, this approach grows exponentially in time with respect to its computational effort. The most common algorithms to address MILP are [11]:

- Branch and bound methods, a binary tree is employed for the representation of the 0 — 1 combinations, the feasible region is partitioned into subdomains systematically, and valid upper and lower bounds are generated at different levels of the binary tree.

- Cutting plane methods, the feasible region is not divided into subdomains but instead new constraints, denoted as cuts, are generated and added which reduce the feasible region until a 0 — 1 optimal solution is obtained.

- Decomposition methods, the mathematical structure of the models is exploited via variable partitioning, duality, and relaxation methods.

- Logic- based methods, disjunctive constraints or symbolic inference techniques are utilized which can be expressed in terms of binary variables.

The case of Mixed integer Nonlinear programming (MINLP) is examined in detail in the following chapter.

1.1.4 Quadratic programming

(QP) examines problems whose objective function has quadratic terms, while the feasible set is specified with linear equalities and inequalities. Quadratic programming is the simplest case of NLP. For specific forms of the quadratic term, this is a type of convex programming. As with NLP there are cases of both convex and nonconvex QP problems. A typical convex QP is formulated [5] as:

$$\begin{aligned} \min_{x} \ & \frac{1}{2}\, x^T P x + q^T x + r \\ & s.t.\ Gx \leq h \\ & Ax = b \end{aligned}$$

where $P \in S_+^n, G \in \mathbb{R}^{m \times n}$

If apart from the objective function, the constrains are in quadratic (QCQP) the formulation is:

$$\min_{x} \frac{1}{2} x^T P_0 x + q_0^T x + r_0$$

$$s.t. \frac{1}{2} x^T P_i x + q_i^T x + r_i \ , \quad i = 1, \cdots, m$$

$$Ax = b$$

1.1.5 Stochastic programming

Stochastic programming problems have constraints or parameters depending on random variables. In contrast to deterministic optimization, real world problems almost invariably include parameters which are unknown at the time a decision should be made. When the parameters are uncertain, but assumed to lie in some given set of possible values, one might seek a solution that is feasible for all possible parameter choices and optimizes a given objective function[23]. The two-stage stochastic programming approach is based on the restriction that decisions should be based on data available at the time and should not depend on future observations. The general formulation for two-staged problems is seen below[23].

$$\min_{x \in X} c^T x + \mathbb{E}[Q(x, \xi)]$$

where $Q(x, \xi)$ is the optimal value of the second-stage problem and $\mathbb{E}[Q(x, \xi)]$ denotes the expected value of the second-stage decision problem. The second-stage problem can be formulated as,

$$\min_{y} q^T y$$

$$s.t. \ Tx + Wy \leq h$$

Here $x \in \mathbb{R}^n$ is the first-stage decision vector, X is a polyhedral set, defined by a finite

number of linear constraints, $y \in \mathbb{R}^m$ is the second-stage decision vector and $\xi = (q, T, W, h)$ contains the data of the second stage problem.

1.1.6 Robust optimization

Similarly to stochastic programming, robust optimization is an attempt to capture uncertainty in the data underlying the optimization problem. Robust optimization aims to find solutions that are valid under all possible realizations of the uncertainties defined by an uncertainty set.

1.2 Convex Analysis

In order to fathom optimization it is necessary to introduce fundamental concepts of convex set and convex function as they are presented in [11]. Basic definitions will be covered and properties of geometrical structures, convex set, convex hull and finally convex function.

Theorem 1.2.1 *Let the vectors $x_1, x_2 \in \mathbb{R}^n$. The **line** through x_1 and x_2 is defined as the set:*

$$\{x|\ x = (1-\lambda)x_1 + \lambda x_2, \lambda \in \mathbb{R}\}$$

Theorem 1.2.2 *Let the vectors $x_1, x_2 \in \mathbb{R}^n$. The **closed line segment** through x_1 and x_2 is defined as the set:*

$$\{x|\ x = (1-\lambda)x_1 + \lambda x_2, 0 \leq \lambda \leq 1\}$$

The **open, closed-open** and **open-closed** line segments can be defined similarly by changing the constraints for λ.

Theorem 1.2.3 *Let the vector $c \in \mathbb{R}^n,\ c \neq 0$ and the scalar $z \in \mathbb{R}$. The* ***open half-space*** *in $\mathbb{R}^n$ is defined as the set:*

$$\{x | c^t x < z, x \in \mathbb{R}^n\}$$

The ***closed half-space*** *in $\mathbb{R}^n$ is defined as the set:*

$$\{x | c^t x \leq z, x \in \mathbb{R}^n\}$$

Theorem 1.2.4 *The* ***hyperplane*** *in $\mathbb{R}^n$ is defined as the set :*

$$\{x | c^t x = z, x \in \mathbb{R}^n\}$$

Theorem 1.2.5 *The intersection of a finite number of closed half-spaces in $\mathbb{R}^n$ is defined as a* ***polytope****. A bounded polytope is called a* ***polyhedron****.*

Theorem 1.2.6 *A set $S \in \mathbb{R}^n$ is said to be* ***convex*** *if the closed line segment joining any two points x_1 and x_2 of the set S,that is, $(1-\lambda)x_1 + \lambda x_2$, belongs to the set S for each $0 \leq \lambda \leq 1$.*

By Definition 1.2.6 convex sets are the following geometrical structures: line, open and closed half-spaces, polytope and polyhedron, all points inside or on the circle. Figure 1.1 is an illustration of convex and nonconvex sets.

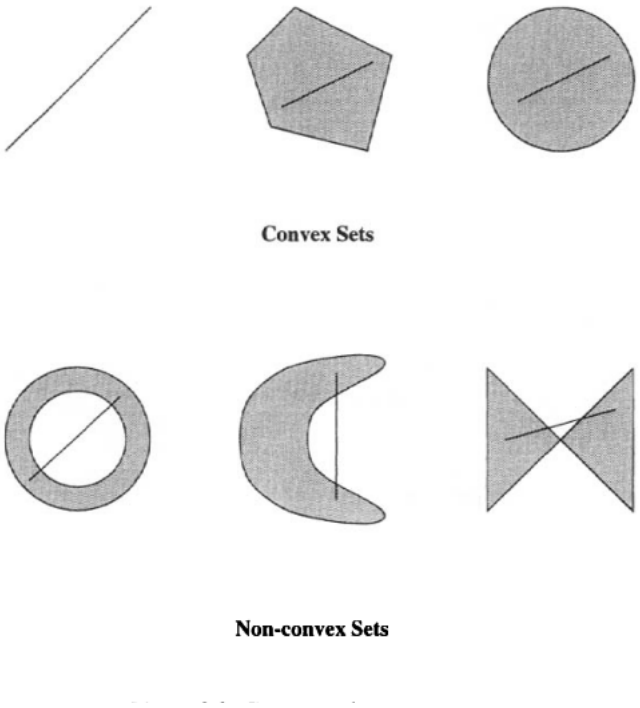


Figure 1.1: Convex and nonconvex sets [11]

Let S_1 and S_2 be convex sets in $\mathbb{R}^n$. Then the properties of the convex sets are :

1. The intersection $S_1 \cap S_2$ is a convex set

2. The sum $S_1 + S_2$ of two convex sets is a convex set

3. The product θS_1 of the real number θ and the set S_1 is a convex set

For every convex and nonconvex set there is a minimum convex set, named the **convex hull**

Theorem 1.2.7 *Let S be a set (convex or nonconvex) in $\mathbb{R}^n$. The **convex hull**, $H(S)$, of S is defined as the intersection of all convex sets in $\mathbb{R}^n$ which contain S as a subset.*

Having introduced the convex set we can now define convex functions.

Theorem 1.2.8 *Let S be a convex subset of $\mathbb{R}^n$, and $f(x)$ be a real valued function defined on S. The function $f(x)$ is said to be convex if for any $x_1, x_2 \in S$, and $0 \leq \lambda \leq 1$, we have*

$$f[(1-\lambda)x_1 + \lambda x_2] \leq (1-\lambda)f(x_1) + \lambda f(x_2)$$

1.3 Review numerical and iterative methods

Finding the optimum solution can be achieved by a variety of methods. The formulation and the complexity of the objective function and the constraints dictates which optimization method should be examined. Selecting the appropriate optimization method is not always apparent, the aforementioned categorization facilitates this process. The optimum can be one point, if we are in search of a global optimum, or it can be more than one points, local optima. According to how strict we are with the optimum selection different methods can be applied.

Numerical methods

The numerical methods are based on the use of necessary and sufficient conditions to define the optimum solution. The numerical methods for optimization require the analytical first and second order derivatives of the objective function, which is demanding in many cases. Based on the Taylor series of the objective function f in a given point $\hat{x} \in \mathbb{R}^n$,

$$f(\hat{x} + \Delta x) - f(\hat{x}) \cong f'(\hat{x}) + f''(\hat{x})\frac{\Delta x^2}{2!}$$

the optimum of P in $\hat{x}$ is met when $\hat{x}$ is a critical point ($f'(\hat{x}) = 0$) and $f''(\hat{x}) \leq 0$ (strong maximum) or $f''(\hat{x}) \geq 0$ (strong minimum). When x is higher dimensional analogous criteria apply.

In the case of constrained optimization problems the Lagrange function is deployed to reformulate P such that the constraints are incorporated in the objective function. The calculated optimum $\hat{x}$ of the Lagrange function ($\mathcal{L}$) guarantees optimality in P.

$$\min_{x,\lambda,\mu} f(x) + \lambda h(x) + \mu g(x)$$

$$\mu \geq 0 \; or \; \mu g(x) = 0$$

where λ and μ are the Lagrange multipliers.

In order to for the Lagrange function to guarantee optimality in P the following conditions, known as the Karush-Kuhn-Tucker (KKT) necessary conditions[3], must apply:

$$\nabla f(\hat{x}) + \lambda \nabla h(\hat{x}) + \mu \nabla g(\hat{x}) = 0$$

$$\mu g(\hat{x}) = 0$$

$$\mu \geq 0$$

Iterative methods

The iterative methods can be divided into direct and indirect methods. Direct methods can examine both smooth and non-smooth functions. Powell method is considered a direct. In Powell method instead of examining all the derivatives to move from one solution point to the other, the principal components of the domain are driving the optimization algorithm. If the function is non-smooth or when the derivatives are unreliable or difficult to compute the appropriate direct methods are the ones who systematically reduce the feasible area by rejecting domains that don't contain the optimum, and polytope methods, such as Simplex method for

LP, which create a polytope and renew the vertex towards the optimum solution. Successive approximation, Newton's method, and other methods for solving nonlinear equations, when applied to an optimization problem, iterate over the equations derived as necessary conditions for an optimal solution. A major disadvantage of this approach is that these iterative techniques may converge only if the initial approximation is sufficiently close to the solution. With these methods only local convergence is guaranteed[18].

The idea behind indirect methods is that, instead of following random or arbitrary directions in the domain space, the direction allowing the greatest improvement of the objective function should be chosen. In this way global convergence, convergence from an arbitrary starting point, often can be insured. We can follow the framework presented in [18], for minimizing a functional f at an initial point x_1, the iterations are guided according to an equation of the form:

$$x_{n+1} = x_n + \alpha_n p_n$$

where α_n is a scalar and p_n is a (direction) vector. Based on the indirect method, the selection of p_n varies, yet once it is chosen the scalar α_n is selected to minimize $f(x_n + ap_n)$, regarded as a function of the scalar α. Usually, things are arranged so that $f(x_n + ap_n) < f(x_n)$ for small positive α. The scalar α_n is often taken as the smallest positive root of the equation

$$\frac{d}{d\alpha} f(x_n + ap_n) = 0$$

Newton's method can be modified for optimization problems to become a rapidly converging descent method, as presented in the following equation:

$$x_{n+1} = x_n - \ \alpha_n [f''(x_n)]^{-1} \ f'(x_n)$$

The most widely used descent procedure for minimizing a functional f is the steepest descent method. Let $\|\cdot\|$ be any norm on $\mathbb{R}^n$. We define, according to [5], a *normalized steepest descent direction* (with respect to the norm $\|\cdot\|$) as,

$$\Delta x_{nsd} = argmin \nabla f(x)^T u | \|u\| = 1$$

where u is the *directional derivative* of f resulting from the first order Taylor approximation:

$$f(x+u) \approx \hat{f}(x+u) = f(x) + \nabla f(x)^T u$$

1.4 The challenge of Global Optimization

Global optimization is the task of finding the absolutely best set of admissible conditions to achieve an objective under given constraints, assuming that both are formulated in mathematical terms. It is much more difficult than convex programming or finding local minimizers of nonlinear programs, since the gap between the necessary (Karush-KuhnTucker) conditions for optimality and known sufficient conditions for global optimality is tremendous [20]. Despite the fact that global optimization is an established research area having many tools/ algorithms for solving optimization problems, not all problems can be addressed to optimality. The effectiveness of these algorithms varies with the particular form of the objective and constraint functions, the number of variables and constraints, and special structure of the problem, such as sparsity (the objective function depends on a small number of variables). Even if the objective

function and constraints are smooth functions, e.g. polynomials, the optimization problem can be still difficult to solve not guarantying an optimum and requiring long computational time.

In some applications of a field of optimization, called embedded optimization, the optimization solution is used to automatically make real-time choices, and even carry out the associated actions, with no (or little) human intervention or oversight [5]. For that reason the optimization algorithms need to provide reliable solutions and solve the problem in a predictable amount of time.

Famous optimization problems such as the traveling salesman problem, the knapsack problem, scheduling problems, protein folding and chemical equilibrium problems are global optimization problems. Many optimization problems such as nonconvex quadratic programming are NP-hard and solving such problems at large scale to global optimality is challenging [2]

1.5 Advanced analytics: a new perspective on global optimization

State-of-the-art solvers of global optimization rely on iterative algorithms. At each iteration new insight is gained about the objective function and the feasible domain driving the algorithm to convergence towards the optimum, however this insight has not been fully exploited by cutting edge techniques. The emerging technologies of machine learning and data analytics are being deployed in an abundance of applications and research areas to accelerate and improve common practices. By embedding data analytics techniques in the present global optimization algorithms underlying information of the problem can be revealed. Incorporating online the information from data analytics to the optimization algorithm can be a great impetus especially

for the problems which state-of-the-art solvers have difficulties to address.

1.6 Motivation

In this study we examine the potential of embedding data analytic's methods in state-of-the-art optimization solvers. Recent advancements in the field of quadratic programming with outer-approximation using cutting planes [2], have revealed a new *separation problem*. By decomposing the original problem and creating a multitude of available low-dimensional cuts there is a need to redefine the cut selection approaches. The current cutting plane selection measures evaluate the cuts based on their contribution with original optimization problem. The selected low-dimensional cuts are proved to be competitive in comparison to the full density cuts especially in the cases of low dimensional QP problems. However, there is no consideration for the complementarity of the selected cuts.

The research procedure is directed by two objectives, define what is the complementarity of the cutting planes and how it can be measured, and incorporate the results from complementarity analysis in the cut selection algorithms to avoid using overlapping cutting planes.

The desirata of this research is to examine the correlation of the cutting planes selected by state-of-the-art algorithms for QP problems and use clustering methods to propose extra selection criteria in the *separation problem*. Of-the-self clustering methods are to be deployed as well as tailor-made methods driven by the insight of the data set. Both approaches aim to improve the reference algorithm in terms of convergence and final solution. There are many different categories of QP problems, in this research we will examine nonconvex quadratic problems which are box and linearly constrained (BoxQP).

1.7 Structure

- Chapter 2 covers a background on global optimization decomposition algorithms addressing Mixed Integer Non Linear Problems(MINLP) and state-of-the-art data analytics methods . The aim of this chapter is to explain in detail necessary concepts that will be used in the following chapters

- Chapter 3 is presents the family of optimization problems that are going to be examined, quadratic programming and box constraint, as well as the cutting edge algorithms proposed to address them. A numerical example is available, to bring forward the key elements of state-of-the-art algorithms. Finally results from the reference study of Balteau-Lugojan, Misener et. al. [2] are presented and discussed.

- Chapter 4 explores the motivation and the proposed methodology using advanced clustering approaches as well as the motivation of the computational experiments. The results and analysis of the computational experiments for k-means and Agglomerative clustering are presented here as well.

- In chapter 5, the affinity norm is introduced and mathematically formulated along with the hybrid approach combining affinity norm and k-means. The results of the corresponding computational experiments of affinity and hybrid algorithms are available in this chapter.

- Chapter 6 summarizes the concluding remarks and future research

Chapter 2

Background and state-of-the-art

This chapter presents the basic concepts of decomposition algorithms, used to address MINLP problems, followed by a review of the most common data analytics algorithms. This chapter provides the background for the present research, the algorithms to be deployed are examined in detail in the following chapters.

2.1 Global optimization- Decomposition algorithms

Addressing QP is very often achieved by the use of decomposition algorithms suitable for MINLP problems. Despite the fact that BoxQP have no integer variables, the same approaches as solving MINLP are being deployed, in this case for only continuous variables, to decompose and approximate the original problem. The nonconvexity of the studied BoxQP requires the relaxation of the the feasible domain into the corresponding convex hull. In each iteration of the algorithm the relaxation is tightened by adding new constraints.

A classic MINLP problem could be expressed as follows,

$$\begin{aligned} &\min_{x,y} f(x,y) \\ s.t.\ &h(x,y) = 0 \\ &g(x,y) \leq 0 \\ &x \in X \subseteq \mathbb{R}^n \\ &y \in Y\{0,1\}^p \end{aligned} \tag{2.1}$$

Solving MINLP shares with MILP the feature that fixing all integer variables leads to a tractabe problem, in this case a convex nonlinear program[20]. The most common techniques for solving MINLP problems include:

2.1.1 Generalized Benders Decomposition

In the pioneering work of Geoffrion (1972)[15] on the Generalized Benders Decomposition GBD, Bender's approach is generalized to not only address linear problems. Two sequences of updated upper (nonincreasing) and lower (nondecreasing) bounds are created that converge within ϵ in a finite number of iterations. The upper bounds correspond to solving sub problems in the continuous variables by fixing the discrete variables, while the lower bounds are based on duality theory. The solution of the primal problem provides information about the upper bound and the Lagrange multipliers associated with the equality and inequality constraints. The master problem is derived via nonlinear duality theory, makes use of the Lagrange multipliers obtained in the primal problem, and its solution provides information about the lower bound, as well as the next set of fixed discrete variables to be used subsequently in the primal problem. As the iterations proceed, it is shown that the sequence of updated upper bounds is nonincreasing, the sequence of lower bounds is nondecreasing, and that the sequences converge in a finite number

of iterations. Originally Geoffrion applied GBD in a maximization problem, Floudas (1995) [11] represents GBD in a minimization problem.

At each iteration k of GBD the binary variables y are fixed to y^k resulting in the NLP primal problem $P(y^k)$.

$$\begin{aligned} &\min_x f(x, y^k) \\ &s.t.\ h(x, y^k) = 0 \\ &g(x, y^k) \leq 0 \\ &x \in X \subseteq \mathbb{R}^n \end{aligned} \qquad \mathrm{P(y^k)}$$

There are two different scenarios for the primal problem, the primal to be feasible (1), and the primal to be infeasible.

1. **Feasible primal**

   If the case of feasible primal problem, the solution provides information on $x^k, f(x^k, y^k)$, which is the upper bound, and the optimal Lagrange multiplier vectors λ^k, μ^k. Given this information we can formulate the Lagrange function as

$$\mathcal{L}(x, y, \lambda^k, \mu^k) = f(x, y) + \lambda^{k^T} h(x, y) + \mu^{k^T} g(x, y)$$

2. **Infeasible primal**

   If the primal is detected by the NLP solver as infeasible, we need to minimize an l_1 or l sum of constraint violations. An l_1 minimization problem can be

$$\begin{aligned} &\min_{x \in X} \sum_{i=1}^{p} \alpha_i \\ &s.t.\ h(x, y^k) = 0 \\ &g_i(x, y^k) \leq a_i, i = 1, 2, \cdots, p \\ &a_i \geq 0, i = 1 \end{aligned}$$

If $\sum_{i=1}^{p} \alpha_i = 0$ then a feasible point has been determined.

Using the Lagrange function and nonlinear duality theory, the master problem **M** is formulated as

$$\begin{aligned} & \min_{y \in Y, \mu_\beta} \ \mu_\beta \\ s.t. \ \mu_\beta \ & \geq \inf_{x \in X} \ \mathcal{L}(x, y, \lambda, \mu) \ \ \forall \lambda, \mu \geq 0 \\ 0 \ & \geq \inf_{x \in X} \ \bar{\mathcal{L}}(x, y, \bar{\lambda}, \bar{\mu}) \ \ \forall (\bar{\lambda}, \bar{\mu}) \in \Lambda \end{aligned} \tag{M}$$

where,

$$\mu_\beta \text{ is a scalar}$$

$$\mathcal{L}(x, y, \lambda, \mu) = f(x, y) + \lambda^T h(x, y) + \mu^T g(x, y)$$

$$\bar{\mathcal{L}}(x, y, \bar{\lambda}, \bar{\mu}) = \bar{\lambda}^T h(x, y) + \bar{\mu}^T g(x, y)$$

$$\Lambda = \{\bar{\lambda} \in \mathbb{R}^m, \bar{\mu} \in \mathbb{R}^p | \bar{\mu} \geq 0, \sum_{i=1}^{p} \bar{\mu} = 1\}$$

2.1.2 Branch and Bound

The branch and bound BB was first proposed by A. H. Land and A. G. Doig in 1960 for discrete linear programming approaches, but cab be extended for MINLP as well. The general BB method is based on the key ideas of separation,relaxation, and fathoming which are outlined in [11].

Let an MINLP problem of the form 2.1 be denoted as (P) and let its set of feasible solutions denoted as $FS(P)$. A set of sub-problems $(P_1), (P_2), \cdots, (P_n)$ is defined as a **separation** of (P) if the following conditions hold:

- A feasible solution of any of the sub-problems $(P_1), (P_2), \cdots, (P_n)$ is a feasible solution of (P), and
- Every feasible solution of (P) is a feasible solution of exactly one of the sub-problems.

An optimization problem, denoted as (RP), is defined as a **relaxation** of the problem (P) if the set of feasible solutions of (P) is a subset of the feasible solutions of (RP),

$$FS(P) \subseteq FS(RP)$$

An important issue in BB is how to generate the relaxation of (P). One way of relaxation is by simply omitting one or several constraints of problem (P). Another way is by setting one or more positive coefficients of binary variables of the objective function, which are still free, equal to zero. Another alternative of generating a valid relaxation is by replacing the integrality conditions on the y variables by $0 \leq y \leq 1$. This type of relaxation results in a linear programming problem, and it is denoted as linear programming relaxation. It is the most frequently used relaxation.

Let (CS) be a candidate problem for solving (P). We would like to determine whether the feasible region of (CS), $(F(CS))$ contains the optimal solution of (P) and find it if it does. A candidate sub-problem (CS) will be considered that has been **fathomed** if one of the following two conditions take place:

1. It can be ascertained that the feasible solution $F(CS)$ cannot contain a better solution than the best solution found so far (i.e., the incumbent);or
2. An optimal solution of (CS) is found

The algorithm of BB, see Figure 2.1 starts by solving the continuous relaxation of the MINLP and subsequently perform an implicit enumeration where a subset of the 0-1 variables is fixed at each node. The lower bound corresponds to the NLP solution at each node and it is used to expand on the node with the lowest lower bound (i.e., breadth first enumeration), or it

is used to eliminate nodes if the lower bound exceeds the current upper bound (i.e., depth first enumeration). If the continuous relaxation NLP of the MINLP has 0-1 solution for the binary variables, then the BB algorithm will terminate at that node. With a similar argument, if a tight NLP relaxation results in the first node of the tree, then the number of nodes that would need to be eliminated can be low. However, loose NLP relaxations may result in having a large number of NLP sub problems to be solved which do not have the attractive update features that LP problems exhibit.

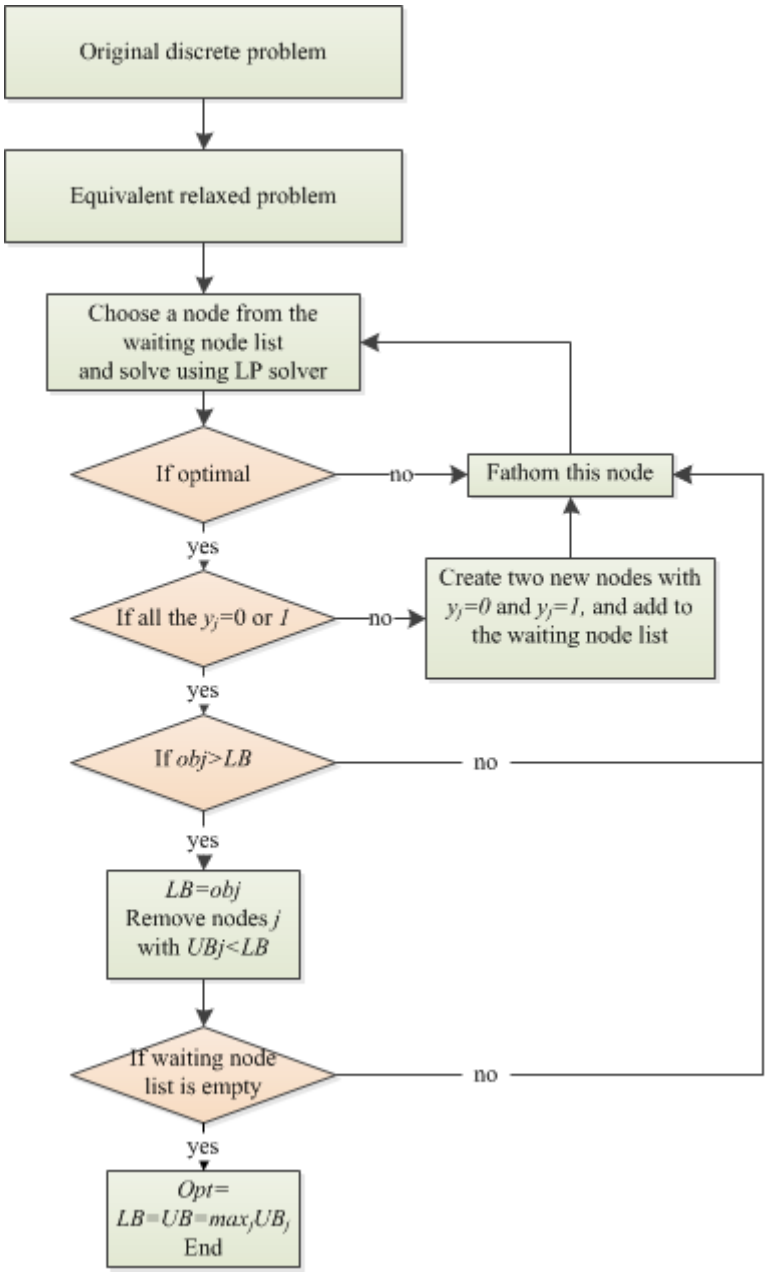


Figure 2.1: The flow chart for Branch and Bound algorithm [14]

2.1.3 Outer Approximation

Outer approximation(OA) was introduced by Duran and Grossmann (1986)[10] for the following class of MINLP problems:

$$
\begin{aligned}
&\min_{x,y} c^T y + f(x) \\
&s.t.\ g(x) + By \leq 0 \\
&x \in X \subseteq \mathbb{R}^n \\
&y \in \subseteq \mathbb{R}^m_+
\end{aligned}
\tag{2.2}
$$

where the nonlinear functions $f : \mathbb{R}^n \to \mathbb{R}$ and those in the vector function $g : \mathbb{R}^n \to R^p$ are assumed to be continuously differentiable and convex on the n-dimensional compact polyhedral convex set $X = \{x|\ x \in \mathbb{R}^n,\ A_1 x \leq \alpha_1\}$; $U = \{y|\ y \in Y$, integer $A_2 y \leq \alpha_2\}$ is a finite discrete set, e.g. the non-negative integer points of some convex polytope, and for most application Y corresponds to the unit hypercube $Y = \{0,1\}^m$. B, A_1, A_2,andc, a_1, a_2 are respectively matrices and vectors of conformable dimensions; the vectors are column vectors unless specified otherwise; finally, some of the rows in B may be the zero row vector, which then defines nonlinear constraints in only the continuous variables.

The aforementioned Generalized Benders Decomposition and Branch and Bound can both address MINLP problems of the form 2.2. However Duran and Grossman[17] introduced OA in order to take advantage of the linearity of the discrete variables and the convexity of the continuous valued functions.

Based on principles of decomposition, outer-approximation and relaxation, OA algorithm effectively exploits the structure of the original problems. The new problem consists of solving an alternating finite sequence of nonlinear programming sub-problems and relaxed versions of a mixed-integer linear master program. The algorithm can also be viewed as a cutting plane method for solving the general convex programming problem where some of the variables are discrete and appear as in problem 2.2. OA shares the same tools of projection, outer-approximation and and relaxation as GBD. Their main difference lies in the the type of outer-

approximation to define the corresponding Master problem. In the outer-approximation algorithm the optimal primal information of the sub-problems is used to define a mixed-integer linear Master problem, in the generalized Benders decomposition method the optimal dual information is used, such that the Master problem corresponds to an initially poorly constrained pseudo-pure integer linear problem [17].

Primal problem

The Primal problem is created by fixing the y variables in 2.2 to a 0-1 combination as y^k, which corresponds to :

$$\begin{aligned} \min_{x} \quad & c^T y^k + f(x) \\ s.t. \quad & g(x) + By^k \leq 0 \\ & x \in X \end{aligned} \tag{2.3}$$

Depending on the allocations of y^k the Primal can be either (1) feasible or (2) infeasible.

1. **Feasible Primal**

   If the primal is feasible at iteration k then its solution on the optimal $x^k, f(x^k)$ provides the upper bound $UBD = c^T y^k + f(x^k)$. Due to convexity of $f(x), g(x)$ we can linearize them around x^k such that:

$$f(x) \geq f(x^k) + \nabla f(x^k)(x - x^k), \forall x^k \in X,$$

$$g(x) \geq g(x^k) + \nabla g(x^k)(x - x^k), \forall x^k \in X,$$

2. **Infeasible Primal**

   If the primal is infeasible we need do reformulate the constraint set in order to get a feasible point. Similarly to GBD, the l_1-minimization will result in:

$$\begin{aligned} & \min_{x \in X} \sum_{j=1}^{p} \alpha_j \\ s.t. \quad & g_j(x) + By^k \leq \alpha_j, \quad j = 1, 2, \cdots, p \\ & \alpha_j \geq 0 \end{aligned}$$

Its solution will provide the corresponding x^t point based on which the constraints can be linearized:

$$g(x) \geq g(x^t) + \nabla g(x^t)(x - x^t), \forall x^t$$

.

Mater problem The key ideas for the derivation of the Master problem in OA are :

- Projection of 2.2 onto the y-space,
- Outer- approximation of the objective function and the feasible region

The final formulation of the Master problem is :

$$\min_{x,y,\mu_{OA}} \quad c^T y + \mu_{OA}$$

$$s.t. \quad \mu_{OA} \geq f(x^k) + \nabla f(x^k)(x - x^k), \quad \forall k \in F,$$

$$0 \geq g(x^k) + \nabla g(x^k)(x - x^k) + By, \quad \forall k \in F,$$

$$x \in X$$

$$y \in Y$$

where $F = \{k : x^k \text{is a feasible solution to the Primal 2.3}$

The Outer Approximation OA can also address problems with nonlinear equalities by implementing equality relaxations (OA/ER).The basic idea in OA/ER is to relax the nonlinear equality constraints into inequalities and subsequently apply the OA algorithm. In this case OA/ER sequences of upper and lower bounds are being created as in GBD, but it has the distinct feature of using primal information, that is the solution of the upper bound problems, so as to linearize the objective and constraints around that point. The lower bounds in OA are based upon the accumulation of the linearized objective function and constraints, around the generated primal solution points[11].

Figure 2.2 graphically illustrates the outer-approximation of a nonconvex objective function, OA dictates that f should be a convex function and not the objective function overall, and the corresponding domain.

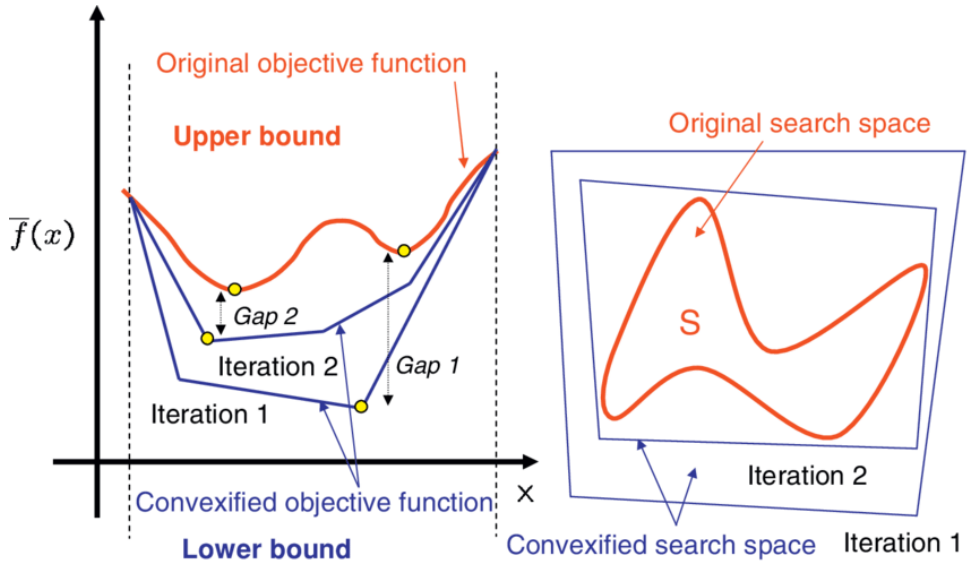


Figure 2.2: Illustration of outer-approximation algorithm[17]

Master problem is a mathematical model repeated over iterations creating a population of underestimators. The created underestimators, following the appropriate formulation can be used as cutting planes.

2.1.4 Cutting plane methods

Cutting plane methods are used for solving MINLP problems. They are based on the the continuous relaxation of the integer variables of the original problem. If the solution of relaxed problem doesn't satisfy an integer solution for the relaxed integer variables, then a linear inequality/cut is added to the constraints representing the feasibility violation. Cutting plane methods do not partition the feasible region into subdivisions, as in branch-and-bound approaches, but they instead works with a single problem, which it refined by adding new constraints. The new constraints successively reduce the feasible region until an integer optimal solution is found. Initially, the branch-and-bound procedures outperformed the cutting-plane algorithm[6]. However, the development of polyhedral theory and the consequent introduc-

tion of strong, problem specific cutting planes led to a resurgence of cutting plane methods in the 1980's, and cutting plane methods are now the method of choice for a wide variety of problems[19].

Finding the appropriate inequality constitutes the well known *separation problem* of the cutting plane selection methods. The difficulty of the *separation problem* varies immensely with the cutting plane being sought [4]. Common cutting planes are *Chvatal-Gomory cutting planes* and strong cutting planes from polyhedral theory [19].

2.1.5 Branch and Cut

The branch and cut algorithms are consisting of a combination of cutting plane and branch and bound algorithms. Branch and Cut(BC) methods work by solving a sequence of linear programming relaxations of the integer programming problem. Cutting plane methods improve the relaxation of the problem to more closely approximate the integer programming problem, and branch-and-bound algorithms proceed by a sophisticated divide and conquer approach to solve problems[19]. BC can be used to address MINLP problems as well relaxed in the appropriate MILP formulation.

The computational experiments of this study are carried using the solver CPLEX, which is based on the branch and cut algorithm.

2.2 Cutting planes as a population- Advance analytics

To address the *seperation problem*, a data driven approach can be deployed in order to build efficiency and acceleration in the process of selecting underestimators/cutting planes. Depending on the formulation of the relaxation and the number of variables the number of underestimators can vary significantly. Even though we could use all of the available underestimators, many of them would provide lower bounds in areas of the domain of the objective function away from the optimum. The population of available underestimators can be examined via a suitable data analytics method to detect the most informative-with respect to the optimum- cuts.

Data Analytics are mathematical techniques using datasets to reveal underlying information.Uncovering patterns provides an insight in the raw dataset. Depending on the nature of the dataset as well as the type of information expected to be extracted, data analytics can be separated in the following techniques.

- **Classification**

  Classification belongs to the Supervised and Semi Supervised techniques since it demands labeled training data consisting of a set of training examples. In classification the training data are labeled to belong in one of the predefined categories. Given a new data point/object, a trained classifier indentifies in which category the object belongs to. Common classification techniques are Logistic regression, Naive Bayes, Support Vector Machines(SVM), Stochastic Gradient Descent and Neural Networks. The applications of classification include spam detection and image recognition.

- **Clustering**

  Clustering techniques perform a similar task as classification but with a great difference. Clustering is an Unsupervised or Semi Supervised technique and no training data set is

used. The aim of clustering is to automatically group similar (in some sense) data points into sets/clusters. Applications span over pattern recognition, image analysis, grouping experimental results and many more. The most common clustering techniques are k-means, Hierarchical clustering(Agglomerative and Divisive). Other clustering algorithms are Density-based spatial clustering of applications with noise (DBSCAN) and Ordering points to identify the clustering structure (OPTICS).

In the following chapters we will implement k-means and Agglomerative clustering algorithms. We will present an outline of these two algorithms as presented by Bramer in [7].

k-means is an exclusive clustering algorithm. Each object is assigned to precisely one of a set of clusters. (There are other methods that allow objects to be in more than one cluster.) For this method of clustering we start by deciding how many clusters we would like to form from our data. We can measure the quality of a set of clusters using the value of an objective function which we will take to be the sum of the squares of the distances of each point from the centroid of the cluster to which it is assigned. We would like the value of this function to be as small as possible. We initialize the centroids from points of the data set or random points in the variable space. The centroid initialization point is of crucial importance, since it yield to a local minimum of the objective function. Advanced k-means algorithm software include initialization algorithms to avoid local minima. Having selected the k centroids, each point of the data set is assigned to a centroid. When all the objects have been assigned we will have k clusters based on the original k centroids but the 'centroids' will no longer be the true centroids of the clusters. Next we recalculate the centroids of the clusters, and then repeat the previous steps, assigning each object to the cluster with the nearest centroid etc.

For a 2-D data set in Figure 2.3 and 2.4 we can observe the assignment of the points in each cluster and the reallocation of the centroids.

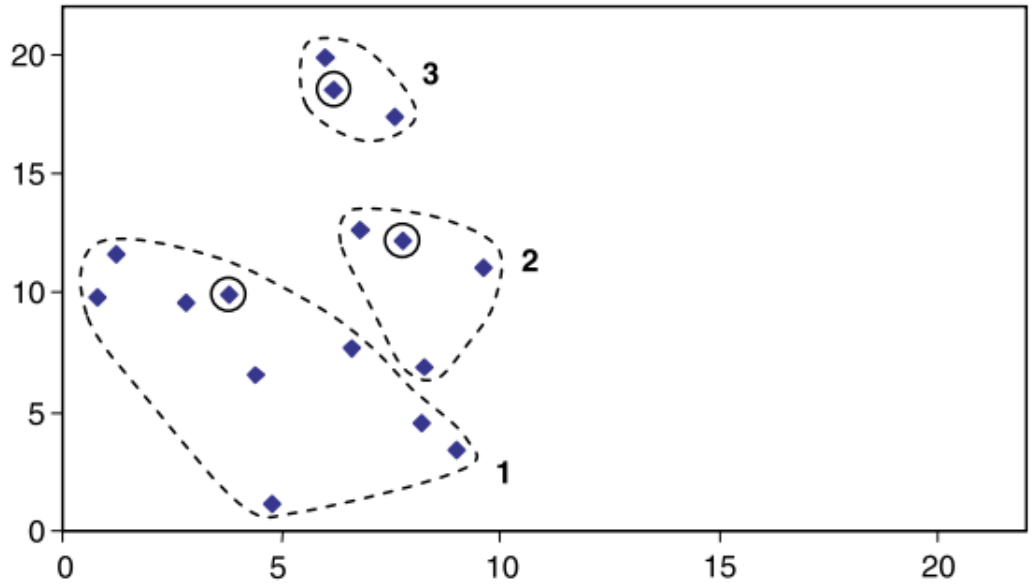


Figure 2.3: Initial cluster of k-means algorithm[7]

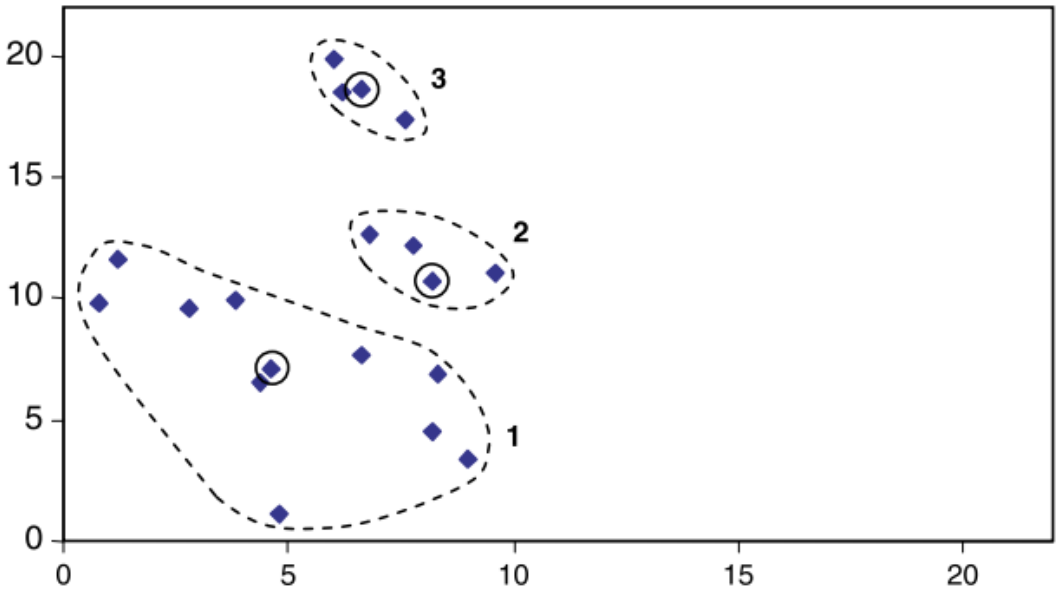


Figure 2.4: Revised clusters of k-means algorithm[7]

Agglomerative clustering

Agglomerative clustering is based on the idea of a reverse dendrogram, each data point is a cluster, clusters are being merged based on the predifined distance metric ,e.g. Euclidean distance. Figure 2.6 represents an example of the dendrogram with eleven objects A, B, C, . . . , K located as shown in Figure 2.5 and we merge clusters on the basis of Euclidean distance.

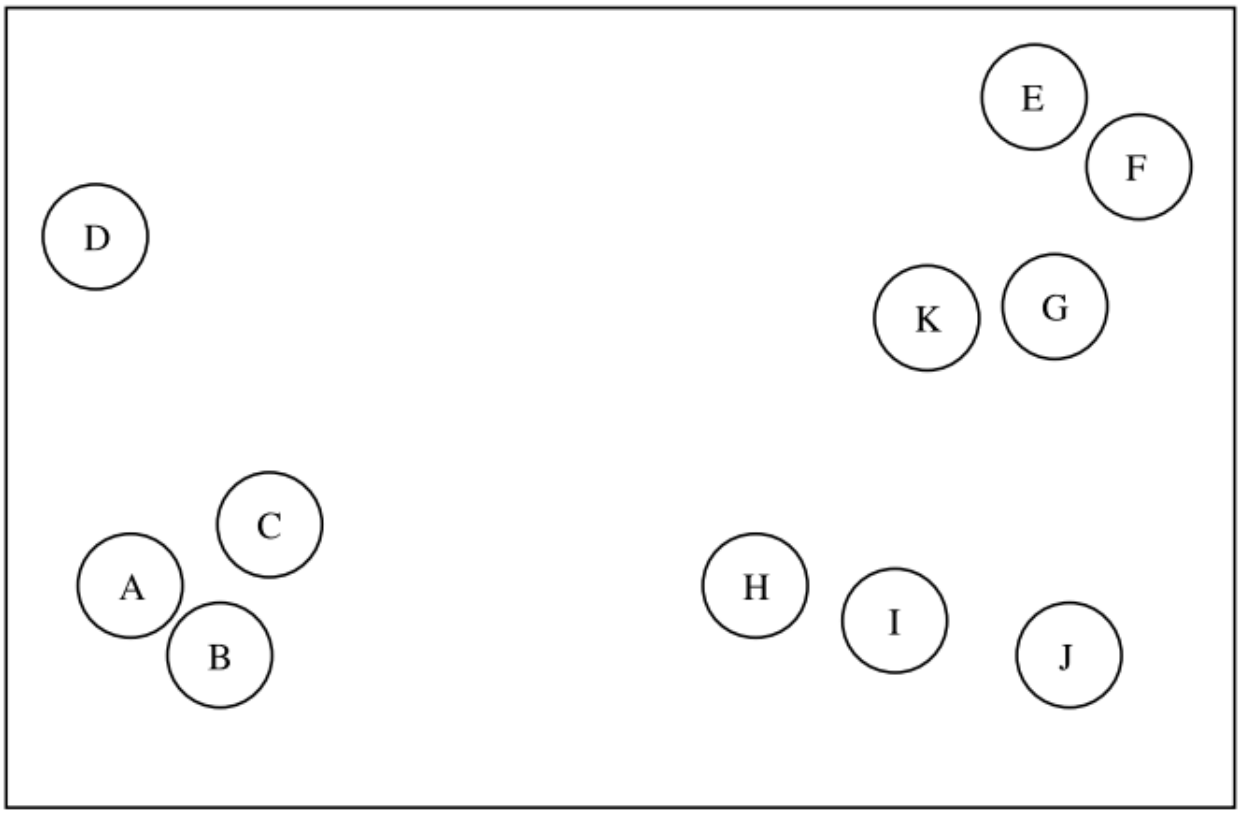


Figure 2.5: Original data[7]

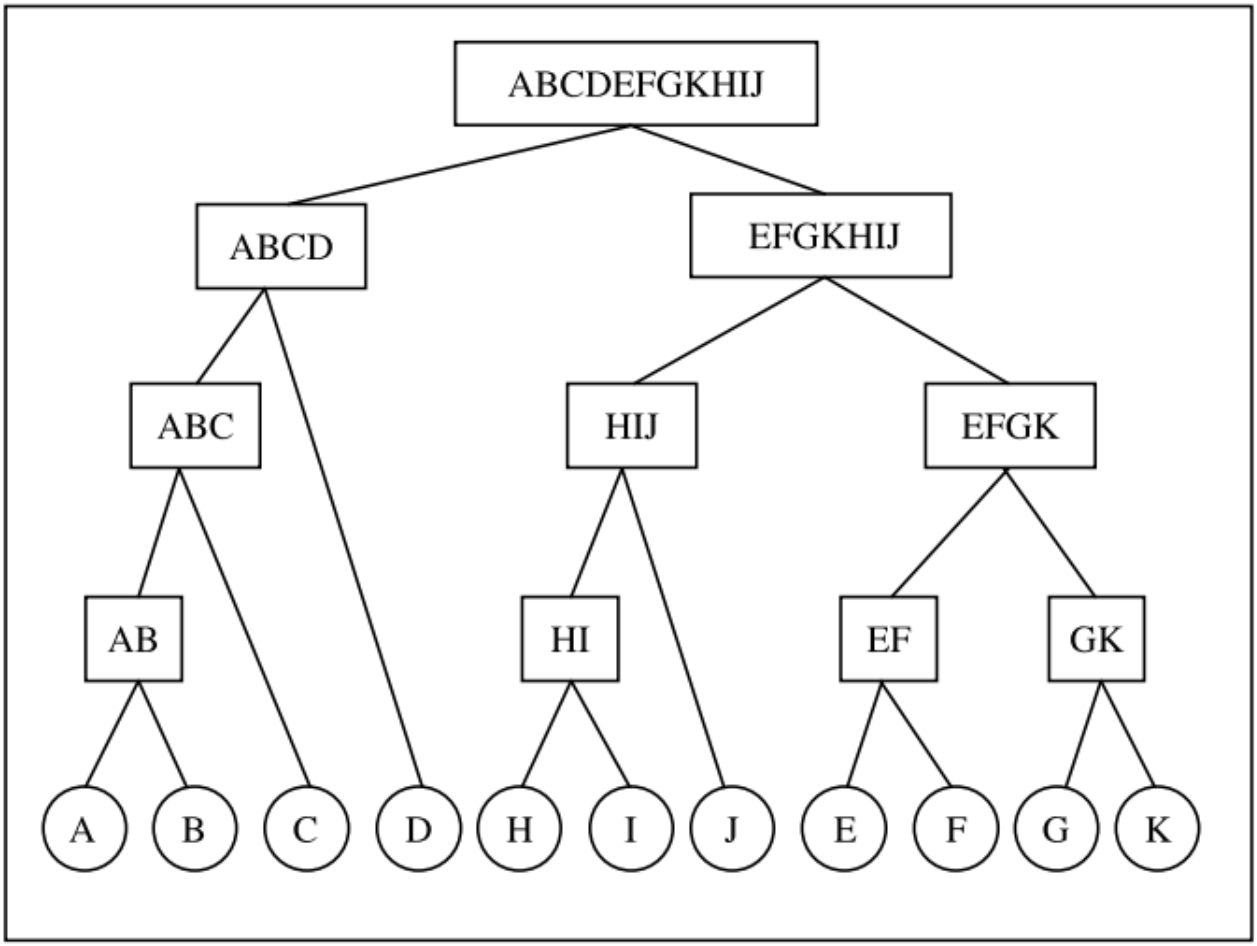


Figure 2.6: A possible dednrogram based on Agglomerative clustering[7]

- **Regression**

  Regression is a Supervised technique aiming to predict a continuous-valued attribute associated with an object. Regression models are predictive models and can be used in an abundance of applications such as drug response and stock prices. Regression models include Logistic regression, Stochastic gradient descent, Naive Bayes, Support vector machines(SVM) and Neural Networks.

- **Dimensionality reduction**

  Dimensionality reduction techniques are used to detect the principal variables of a variable set. Dimensionality reduction techniques are useful in problems such as visualization. The

most common techniques are Principal components analysis (PCA), Non-negative matrix factorization (NMF) and Linear discriminant analysis (LDA).

Chapter 3

Pilot application: Box quadratic problems

The research of this study attempts to improve SoA algorithms for solving BoxQP. In the following chapter SoA algorithms are being presented as well as an numerical example of how they are being implemented. The test set of BoxQP studied in [2], will be the test set of this research as well, is composed of 99 BoxQP instances.

The 99 instances come from the following sources: (i) 54 instances with sizes $20 \leq N \leq 60$ generated by Vandenbussche and Nemhauser (2005)[25]; (ii) 36 instances with sizes $70 \leq N \leq 100$ generated by Burer and Vandenbussche (2009)[9]; and (iii) 9 instances with $N = 125$ generated by Burer (2010)[8].

3.1 Solving BoxQP via RLT and SDP relaxations

The optimization problem is considered a nonconvex quadratic problem that is box and linearly constrained, i.e.

$$z_{qp} = \min_{x}\{x^T Q x + c^T x \mid Ax \leq b,\ x \in [0,1]^N\}$$

with an N-variable vector x, $A \in \mathbb{R}^{p\times N}$ and $Q \in \mathbb{R}^{N\times N}$ assumed to be an indefinite matrix. For the problem to be solved a series of reformulations and relaxations are taking place as proposed in [24]. Each quadratic term $x_i x_j$ is replaced by a new variable X_{ij}. The lifted variables $X_{ij} \forall i, j$ form the matrix$X = xx^T$ and let $Q \bullet X = Tr(Q^T X) = \sum_{i,j} Q_{ij} X_{ij}$, representing the Frobenius inner product (applied to pairs of either matrices or vectors with the same dimensions). Then the z_{qp} is lower-bounded by,

$$z_{qp}(\mathcal{B}) := \min_{x,X}\{Q \bullet X + c^T x \mid Ax \leq b,\ x \in [0,1]^N and\ (x,X) \in \mathcal{B}\},$$

parametric on any convex set $\mathcal{B}$ that adds valid constraints to the basic lifted formulation of the quadratic problem.

Let $G(V,E)$ denote the sparsity pattern graph introduced by matrix Q (linking lifted X variables) where set V and edge E are defined as

$$V = \{1, 2, ..., N\}, \qquad E = \{\{i,j\} \in V \times V \mid i > j, Q_{ij} \neq 0\}$$

The relaxation of nonconvex $X = xx^T$ to $X \geq xx^T$, or equivalently $\begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \geq 0$ results in the $\mathcal{S}$ semidefinate relaxation (SDP) of the quadratic problem with a positive semidefinite (PSD) restriction [24, 22].

$$\mathcal{S} := \left\{ (x,X) \;\middle|\; \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \geq 0,\ X_{ii} \leq x_i\ \forall i \in V \right\}.$$

The semidefinate relaxation $\mathcal{S}$ is augmented by the reformulation-linearization technique (RLT) or the McCormick bounds $\mathcal{M}$ [1]. By taking into account only the four original bounds $x_i - l_i \geq 0, x_i - u_i \leq 0, x_j - l_j \geq 0, x_j - u_j \leq 0$, we get the MacCormick $\mathcal{M}$ inequalities

$$\mathcal{M} := \left\{ (x, X) \middle| \forall i, j \in V \text{ and } \{i, j\} \in E : \begin{array}{l} X_{ij} \geq l_i x_j + l_j x_i - l_{lj} = 0, \\ X_{ij} \geq u_i x_j + u_j x_i - u_i u_j = x_i + x_j - 1, \\ X_{ij} \leq l_i x_j + u_j x_i - l_i u_j = x_i, \\ X_{ij} \leq u_i x_j + l_j x_i - u_i l_j = x_j. \end{array} \right\}$$

In the initial round of optimization ,the positive semidefinite constraint is omitted. Let the solution of the initial round be X^*, x^*; to evaluate if the positive semidefinite constraint is guaranteed, eigendecomposition is performed on $\begin{bmatrix} 1 & x^{*T} \\ x^* & X^* \end{bmatrix}$,with t the negative eigenvalues of the matrix. If $t = O$ then PSD is met, if $t \geq 1$ then PSD is violated. For every t the corresponding eigenvector v_k is used to generate the violated cutting planes $v_k^T \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} v_k \geq 0, \ \forall k \in 1, .., t$ which are used as PSD constraints in the following round. We introduce the EigenCut(ρ),

$$v_k^T \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} v_k \geq 0, \ \forall k \in 1, .., t \qquad \text{EigCut}(\rho)$$

Qualizza et. al. (2012)[22], observed that the generated cuts are few in number, one cut per negative eigenvalue, and also very dense, i.e. almost all entries in v_k are nonzero, causing the reoptimization of the linear relaxation to slow down. For that reason they introduced a heuristic for the sparcification of the PSD cuts.

3.2 Sparsification prior to cut generation

To overcome the problem of few dense cuts, Baltean-Lugojan, Misener et. al. [2], introduced low-dimensional approach leading to lighter linear relaxations.

With $\mathcal{P}$ denoting the power set of the vertex set V and $\rho \in \mathcal{P}(\rho \subseteq V)$ any arbitrary index subset, let $x_\rho \in \mathbb{R}^{|\rho|}$ the vector slice of x and $X_\rho \in \mathbb{R}^{|\rho|\times|\rho|}$ the submatrix slice of X. For any subset of $\mathcal{P}$ the following corresponding semidefinite relaxation is introduced,

$$(\forall \mathcal{F} \subseteq \mathcal{P}) \;\; \mathcal{S}(\mathcal{F}) := \left\{ (x, X) \middle| \forall \rho \in \mathcal{F} : \begin{bmatrix} 1 & x_\rho^T \\ x_\rho & X_\rho \end{bmatrix} \geq 0, \; X_{ii} \leq x_i \; \forall i \in \rho \right\}$$

A fixed cardinality $n(1 \leq n \leq N)$ is imposed upon $\mathcal{P}$ such that :

$$\mathcal{P}_n := \{\rho \in \mathcal{P} |\ |\rho| = n\}, \; with \; |\mathcal{P}_n| = \binom{N}{n}$$

A *separation problem* arises, since there are $\binom{N}{n}$ available inequalities to create cutting planes.In [2], the *separation problem* is addressed with one of the following strategies: feasibility, optimality and combined selection.

If **feasibility** strategy is selected, the eigenvalues of $\begin{bmatrix} 1 & \tilde{x_\rho} \\ \tilde{x_\rho}^T & \tilde{X}_\rho \end{bmatrix}$ are calculated and for each ρ sub-problem the minimum eigenvalue is selected. The sub-problems are then being ranked with ascending eigenvalues, $\lambda_{min}(\rho)$ (the sub-problems with the more negative eigenvlaues first).

Optimality strategy represents the improvement of the objective function resulted by the selection of a specific cut. The objective improvement is defined as $ObjImprX(\rho) = f^*(X_\rho^*|\tilde{x_\rho}) - f(\tilde{X}_\rho) = I_X(\rho)$

$$\forall S \in \mathcal{P}_n \begin{cases} f_S^*(X_S^*|\tilde{x_S}) = \underset{X_S}{min} Q_S \bullet X_S \\ s.t. \begin{bmatrix} 1 & \tilde{x_S} \\ \tilde{x_S}^T & X_S \end{bmatrix} \geq 0, \ X_{ii} \leq \tilde{x_i} \ \forall i \in S \end{cases}$$

For high dimensional problems and small dimensional cuts ($n \geq 3$) the optimization problems need to be solved are by SoA solvers results in significant time delays. It takes 5s on average to solve such a simple optimization problem but given 100 variables and 3-D cuts the time needed is $\binom{100}{3} \times 5s \simeq 45h$ just for one cut round. In order to overcome this obstacle Baltean-Lugojan, Misener et. al. [2], introduced the ***fast estimator*** $f^*(X_S^*|\tilde{x_S}) \approx \hat{f}_n^*(Q_s, \tilde{x_s})$. The ***fast estimator*** is computed by a trained neural network and takes the complexity of cut selection offline. $I_X(\rho)$ is approximated by $\hat{I}_X(\rho)$.

$$\hat{I}_X(\rho) \approx \hat{I}_X(\rho) = \hat{f}_n^*(Q_s, \tilde{x_s}) - Q_s \bullet \tilde{X}_s$$

If **combined** strategy is selected, the combined measure is

$$C(\rho) = \begin{cases} \hat{I}_X(\rho) + T, & \text{if } \hat{I}_X(\rho) > 0 \text{ and } \lambda_{min}(\rho) < 0 \\ -\lambda_{min}(\rho) & \text{otherwise} \end{cases},$$

where T is an arbitrary large positive number.

Algorithm 8 [2]sets the framework to outer-approximate $\mathcal{B} + \mathcal{S}$ given any $\mathcal{B}$ linear base relaxation and $\mathcal{B} \subseteq \mathcal{P}_n$ for small $n \leq 5$.

Algorithm 1: Iterative SDP outer-approx. with cut selection/generation based on an ordering

input :
-current base LP relaxation of $\mathcal{B}$ of QP, either fully added from the start,i.e. $\mathcal{M}$ or separates iteratively at each cut round;
-decomposed SDP relax. $\mathcal{S}(\mathcal{F})$ to outer-approx., where $\mathcal{B} \subseteq \mathcal{P}_n$ with small n;
-incumbent LP solution $(\tilde{x}, \tilde{X})$;
- selection strategy/ordering metric $M(\rho)\forall\rho \in \mathcal{F}$ at $(\tilde{x}, \tilde{X})$ e.g. $\hat{I}_X(\rho), -\lambda_{min}(\rho), \mathcal{C}(\rho)$ etc.;
-selection size, i.e. a fixed % of $|\mathcal{F}|$or a fixed number of cuts;
-number of cut rounds $\mathcal{R}$ (set to 20);
-termination criteria, if active terminate on an improvement between to consecutive cut rounds of$\leq$ 0.01% of the gap closed overall so far from the $\mathcal{M}$ bound;

output : Polyhedral outer-approximation that lower bounds $z(\mathcal{B} + \mathcal{S}(\mathcal{F}))$ and SDP relax.$z(\mathcal{B} + \mathcal{S})$;

for $\mathcal{R}$ *cut round if termination criteria not met* **do**
  Sort $\mathcal{F}$ by descending $M(\rho)\forall\rho \in \mathcal{F}$ at current$(\tilde{x}, \tilde{X})$;
  for *top* ρ *sub-problems in sorted* $\mathcal{F}$ *within selection size* **do**
    if $-\lambda_{min}(\rho) < 0$ *(viol. PSD condition for* $\begin{bmatrix} 1 & \tilde{x_\rho} \\ \tilde{x_\rho}^T & \tilde{X}_\rho \end{bmatrix}$*)* **then**
      $\mathcal{B} = \mathcal{B} \cup$ {new EigCut (ρ) based on $-\lambda_{min}(\rho)$}
  Resolve (warm-start) new LP relaxation $\mathcal{B}$ that includes added cuts;
  Update current incumbent solution $(\tilde{x}, \tilde{X})$;
Last obtained $z(\mathcal{B})$ lower bounds $z(\mathcal{B} + \mathcal{S}(\mathcal{F}))$ and $z(\mathcal{B} + \mathcal{S})$;

3.3 Numerical example

In this section a numerical example is presented in order to fathom outer-approximation with cut selection of BoxQP instances. The algorithm will be examined in to variations: (i) first without cut sparsification as in Sherali and Fraticelli (2002) [24]; (ii) sparsification prior to cut generation as proposed by Baltean-Lugojan, Misener et. al. [2].

Let N=5 the number of variables , $x^T = [x_1, x_2, x_3, x_4, x_5]$ and $Q = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 6 & 8 & 10 \\ 3 & 6 & 9 & 12 & 15 \\ 4 & 8 & 12 & 16 & 20 \\ 5 & 10 & 15 & 20 & 25 \end{bmatrix}$,

$c^T = [1, 2, 3, 4, 5]$

The initial problem can be formulated as:

$$\min_{x} \; x^T Q x + c^T x \; s.t. \; Ax \leq b, x \in [0,1]^N \tag{3.1}$$

With the relaxations described in previous section eq.3.1 takes the final formulation of

$$\min_{x,X} \; Q \bullet X + c^T x \tag{3.2}$$

$$Ax \leq b, \tag{3.3}$$

$$X_{ij} - x_i - x_j \geq -1 \tag{3.4}$$

$$X_{ij} - x_i \leq 0 \tag{3.5}$$

$$X_{ij} - x_j \leq 0 \tag{3.6}$$

$$X \geq xx^T \tag{3.7}$$

$$x \in [0,1]^5 \tag{3.8}$$

$$X \in [0,1]^{5\times 5} \tag{3.9}$$

(i) Cut without sparsification

Round=0

1. In the initial round only RLT constraints are active ineq.3.3 -3.6 and 3.8,3.9. The SDP relaxation of ineq.3.7 is not used yet. We solve 3.2 under the active constraints and get the incumbent solution $\tilde{x}, \tilde{X}$, e.g. $\tilde{x}^T = [0.1, 0.2, 0.3, 0.4, 0.5]$ and $\tilde{X} = \begin{bmatrix} 0.2 & 0.3 & 0.5 & 0.7 & 0.8 \\ 0.3 & 0.6 & 0.9 & 0.1 & 0.6 \\ 0.5 & 0.9 & 0.2 & 0.5 & 0.4 \\ 0.7 & 0.1 & 0.5 & 0.9 & 0.3 \\ 0.8 & 0.6 & 0.4 & 0.3 & 1 \end{bmatrix}$

2. We then examine if the PSD constraint for ineq.3.7 is satisfied by calculating the eigenvalues of $\begin{bmatrix} 1 & \tilde{x}^T \\ \tilde{x} & \tilde{X} \end{bmatrix}$ The computed eigenvalues are $\lambda^T = [-0.66, -0.45, 0.47, 0.76, 0.88, 2.29]$, there are two negative eigenvalues($\lambda_1 \leq 0, \lambda_2 \leq 0$) and thus two PSD violations. Let $v_1, v_2 \in \mathcal{R}^{6\times 6}$ be the eigenvectors of λ_1, λ_2, then the created cutting planes are,

$$v_1 \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} v_1^T \geq 0 \tag{3.10}$$

$$v_2 \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} v_2^T \geq 0 \tag{3.11}$$

Round=1

3. Add the EigenCuts constraints ineq.3.10 and 3.11 in the optimization problem and solve again. If there is no negative eigenvalue terminate,otherwise create EigenCuts for the violated constraints.

(i) Sparsification prior to cut generation

1. Step 1 is the same in both approaches for solving 3.2 with RLT constraints and calculating $\tilde{x}, \tilde{X}$.

2. Consider all smaller subsets $S \subset \overline{1, N}$ and check for PSD violation. Let $n = 3$, and thus creating 3-D cuts. For feasibility measure : Compute eigenvalues of $\begin{bmatrix} 1 & \tilde{x_S}^T \\ \tilde{x_S} & \tilde{X}_s \end{bmatrix}, \forall S \in \mathcal{P}_\ni$. For every sub-problem only the most negative eigenvalue is considered. Let $S = 1, 2, 3$ then $\tilde{x_S}^T = [0.1, 0.2, 0.3]$. The eigenvalues are $\lambda^T = [-0.59, 0.03, 0.82, 1.7]$

   Order the sub-problems based on the most negative eigenvalues (feasibility measure).

   Round=1

3. Add the EigenCut constraints for the top violated sub-problems based on eigenvalue ordering and resolve the optimization problem. Enforce termination criteria e.g. maximum number of rounds, threshold in convergence, no negative eigenvalues.

3.4 Benchmark performance

Figure 3.1 corresponds to **Figure 9** of the reference study [2]. There are two perspectives to examine Fig. 3.1:

1. Comparison of $\%M$ to $M + S_3$ gap closed with $M + S$ for different dimensionality and sparsity

2. Comparison of the different selection measures for different dimensionality and sparsity

The $\mathcal{M} + S$ boundary corresponds to the final solution of the optimization problem if full dense cuts are used while $\mathcal{M} + S_3$ corresponds to the solution for 3-D cuts. The $\mathcal{M} + S$ boundary resulted from fully dense cuts is replaced with $\mathcal{M} + S_3$ to examine low-dimensional cuts in order to accelerate the reoptimization of algorithm. Overall we observe that the gap between $\mathcal{M} + S$ and $\mathcal{M} + S_3$ is greater in the cases of high density problems Fig.3.1b and 3.1d and in small densities the gap widens with the increase of variables 3.1a and 3.1c. The results indicate that low-dimensional cuts can replace adequately low-dimensional(40 variables) and low density (30%) problems and is the dimensionality and density increase their performance deteriorates.

In the following chapters we will compare the results attained with $\mathcal{M} + S_3$ which will be the *convergence limit*.

In all of the cases the results reveal that the proposed measure of $\hat{I}_X(\rho)$ used in Algorithm 1 is resulting in significant improvements with respect to convergence compared to $-\lambda_{min}(\rho)$. However feasibility selection reaches better final bounds. Random selection for Fig.3.1.b and Fig.3.1.d suggests that in high density problems every added cutting plane is driving the optimization algorithm. In this case random selection could be an indication that there is complementarity in the selected cutting planes for high density problems by feasibility and optimality measure. In low density problems the population of available cutting planes is significantly smaller and thus random selection can not be as disperse as in the case of high density.

Even though there are indications of complementarity, the relationship between the selected cutting planes and how they map the solution space have not been addressed. Observing the data another question arises, could a more efficient cut selection strategy overcome the bound of $M + S_3$ and result in a greater gap closure with $M + S$? The $M + S_3$ boundary is the result of feasibility selection for 40 rounds and 5% ρ selected, overcoming $M + S_3$ in 20 rounds would result in a significant acceleration of the optimization algorithm.

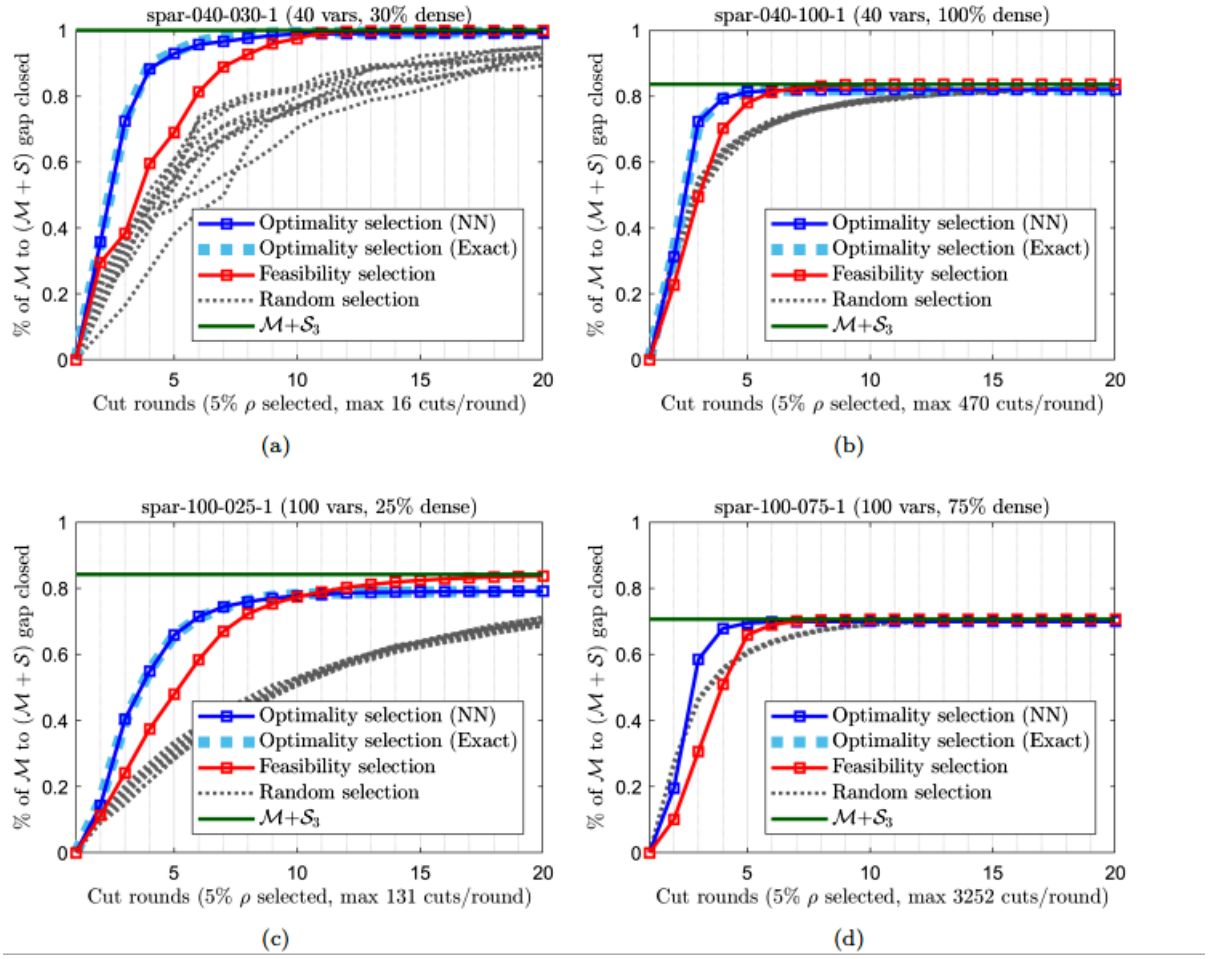


Figure 3.1: Bound convergence (root node) for outer-approximating the M+S3 relaxation starting from $\mathcal{M}$ for BoxQP instances with different sizes and/or densities. Algorithm 1 implements the cutting plane selection/ordering strategies across 20 cut rounds, with selection size $5\% \times |P_3|$ (exact number shown for each instance). Each x-axis cut round shows the percentage of gap closed between the relaxations $\mathcal{M}$ and $\mathcal{M}+S$, and the $\mathcal{M}+S_3$ bound targeted by the outer-approximation is shown in green [2]

Chapter 4

The *separation problem* through advanced data analytics

The intention of this chapter is to used advanced data analytics to address the *separation problem*. We are using the population of the available sub-problems in order to detect areas with congested cutting planes. By using clustering methods we attempt to group the cutting planes based on similarity patterns and use this clustering to achieve a more eclectic selection for the *separation problem*. The clustering is performed online as the optimization algorithm progresses.

In detail, in order to make the Master problem efficient we need to find the best underestimators/cutting planes which will allow faster convergence of the optimization algorithm. We are imposing an extra sorting criterion for the inequality constraints added in the Master problem, not only should they be highly ranked based on eigenvalues(feasibility) or the improvement of the objective function (combined) but they should also be independent to each other. The *separation problem* is redirected from inequality independence, to sub-problem independence

and finally to $\tilde{x_\rho}$ independence, since each cutting plane is created form a sub-problem.

They rationale here is that if one cutting plane is selected, there has to be a guarantee that any additional selected cutting plane added to the list of constraints provides new information for the problem. Both may be informative enough however the information they provide may be overlapping. In order to evaluate the cutting planes we will examine the complementarity of the slices of $\tilde{x_\rho}$ they are composed of. Having dominating variables, variables that are active in the majority of the selected cutting planes, could result in slowing down the optimization algorithm and poorly mapping the variable space.

To achieve a high variance in the active variables of the selected cuts of-the-self clustering algorithms of k-means and Agglomerative clustering are being deployed using the Euclidean norm. The algorithms proposed for clustering are explained in detail with the perspective of improving the aforementioned Algorithm 8[2]. For the purpose of the research the cardinality of the examined sub-sets is set to $|\rho| = 3$ and the selection size is 100 cuts/round.

For feasibility selection the population is comprised of all the sub-problems with negative eigenvalues ($\lambda_{min}(\rho) < 0$) and in optimality selection of all sub-problems with positive objective improvement ($\hat{I}_X(\rho) > 0$), respectively for combined selection. In addition different criteria have been implemented as to evaluate the representative element of each cluster based on the selected number of clusters. Round of Experiments 1 examines k-means algorithm while Round of Experiments 2 Agglomerative clustering.The methodology of this approach can be found in Algorithm 2.

In the first section of this chapter we are examining k-means method and how is embedded with the optimization algorithm, followed by the results of Round of Experiments 1. In section 4.3 we are using Agglomerative clustering followed by a comparison of the results of

Agglomerative clustering, Round of Experiments 2, and k-means.

4.1 k-means

The k-means algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares (see below). In a mathematical formulation k-means divides a set of N samples X into K disjoint clusters C, each described by the mean μ_j of the samples of the cluster. The means are commonly called centroids and they are not necessarily points from X, although they live in the same space.

The k-means algorithm aims to choose centroids that minimise the inertia, or within-cluster sum-of-squares criterion:

$$\sum_{i=0}^{n} \underset{\mu_j \in C}{min}(\|x_i - \mu_j\|^2)$$

k-means algorithm is derived from the scikit-learn python package [21] defining the number of clusters for default settings.

Algorithm 2: Iterative SDP outer-approx. with cut selection/generation based on of-the-self clustering methods

input :
-current base LP relaxation of $\mathcal{B}$ of QP, either fully added from the start,i.e. $\mathcal{M}$ or separates iteratively at each cut round;
-decomposed SDP relax. $\mathcal{S}(\mathcal{F})$ to outer-approx., where $\mathcal{B} \subseteq \mathcal{P}_n$ with small n;
-incumbent LP solution $(\tilde{x}, \tilde{X})$;
- selection strategy/ordering metric $M(\rho) \forall \rho \in \mathcal{F}$ at $(\tilde{x}, \tilde{X})$ e.g. $-\lambda_{min}(\rho), \mathcal{C}(\rho)$ etc.;
-selection size, i.e. a fixed % of $|\mathcal{F}|$or a fixed number of cuts (set to 100);
-number of cut rounds $\mathcal{R}$ (set to 20);
-termination criteria, if active terminate on an improvement between to consecutive cut rounds of$\leq$ 0.01% of the gap closed overall so far from the $\mathcal{M}$ bound;
-of-the-self clustering technique (k-means or Agglomerative clustering);
- number of clusters k

output : Polyhedral outer-approximation that lower bounds $z(\mathcal{B} + \mathcal{S}(\mathcal{F}))$ and SDP relax.$z(\mathcal{B} + \mathcal{S})$;

for $\mathcal{R}$ *cut round if termination criteria not met* **do**
  Cluster all elements in $\mathcal{F}$;
  for *Every cluster* **do**
    Sort all elements in cluster by descending $M(\rho) \forall \rho \in \mathcal{F}$ at current$(\tilde{x}, \tilde{X})$;
    if $k \geq$ *selection size* **then**
      Create EigCut (ρ) based on $-\lambda_{min}(\rho)$ for the top (1st) sorted element in cluster;
      Let the set Eg containing all the selected eigencuts then $Eg = Eg \cup \{EigCut(\rho)\}$;
    if $k \leq$ *selection size* **then**
      for *top selection size/k sub-problems in cluster* **do**
        Create EigCut (ρ) based on $-\lambda_{min}(\rho)$ and $Eg = Eg \cup \{EigCut(\rho)\}$
  Sort EigCut (ρ) in Eg based on $M(\rho)$;
  for *top EigCut (ρ) in Eg within selection size* **do**
    $\mathcal{B} = \mathcal{B} \cup \{$ EigCut (ρ)$\}$
  Resolve (warm-start) new LP relaxation $\mathcal{B}$ that includes added cuts;
  Update current incumbent solution $(\tilde{x}, \tilde{X})$
Last obtained $z(\mathcal{B})$ lower bounds $z(\mathcal{B} + \mathcal{S}(\mathcal{F}))$ and $z(\mathcal{B} + \mathcal{S})$;

4.2 Round of Experiments 1: k-means - Results

The aim of the computational experiments are to evaluate the:

- Existence of complementarity in selected cutting planes with previous algorithm (Rf)
- Correlation between the number of clusters and complementarity, for of-the self clustering approaches the number of clusters k is set to k=10,100,500. The number of clusters are selected as to be comparable to the number of cuts added per round(=100 cuts/round)

- Dimensionality and sparsity dependence with complementarity. Medium and high dimensional problems are examined since the decomposition of the SDP relaxation has a greater impact on high dimensional problems. Different densities are examined in order to see how the complexity of the problem affects cutting plane selection

- Cutting plane selection measure (feasibility:Rf1, combined: Rf2) dependence with complementarity and synergy of the proposed clustering algorithms with SoA approaches

For all the computational experiments $n = 3$, which is the dimensionality of the created cutting planes. The computational experiments where conducted for varying the dimensionality and sparsity of the original optimization problems ($N =, 70, 100$, density= 25%, 50%, 75%). Number of iterations/rounds of the optimization algorithm is set to 20 and added cuts per round (selection size) are 100 cuts/round.The cut selection measure is either feasibility or combined. Setting both the number of rounds and the number of cuts per round allows us to compare the proposed clustering approaches with Rf within the same framework. The potential of the clustering algorithms is only compared to Rf. The results of all experiment rounds are in Appendix B.

The *convergence limit* is the final solution achieved for feasibility measure,when cut rounds are set to 40 and the cut/round are set to 5% of the available sub-problems, defined in [2]. The results are evaluated based on the convergence of the proposed algorithms and the final solution.

The computational experiments where carried in python 3.5 using cplex 12.8 python API solver and scikit-learn v0.2 package for k-means and Agglomerative clustering.

In the first round of computational experiments Algorithm 2 is implemented for k-means clustering algorithm. The parameter is the number of clusters selected (k=10,100,500). Figure

4.1 illustrates the results for different dimensionality : Fig.4.1a with 70 variables and Fig.4.1b with 100 variables, and for different density: Fig.4.1b for 100 variables and 50% density, Fig.4.1c for 25% density and Fig.4.1d for 75% density.

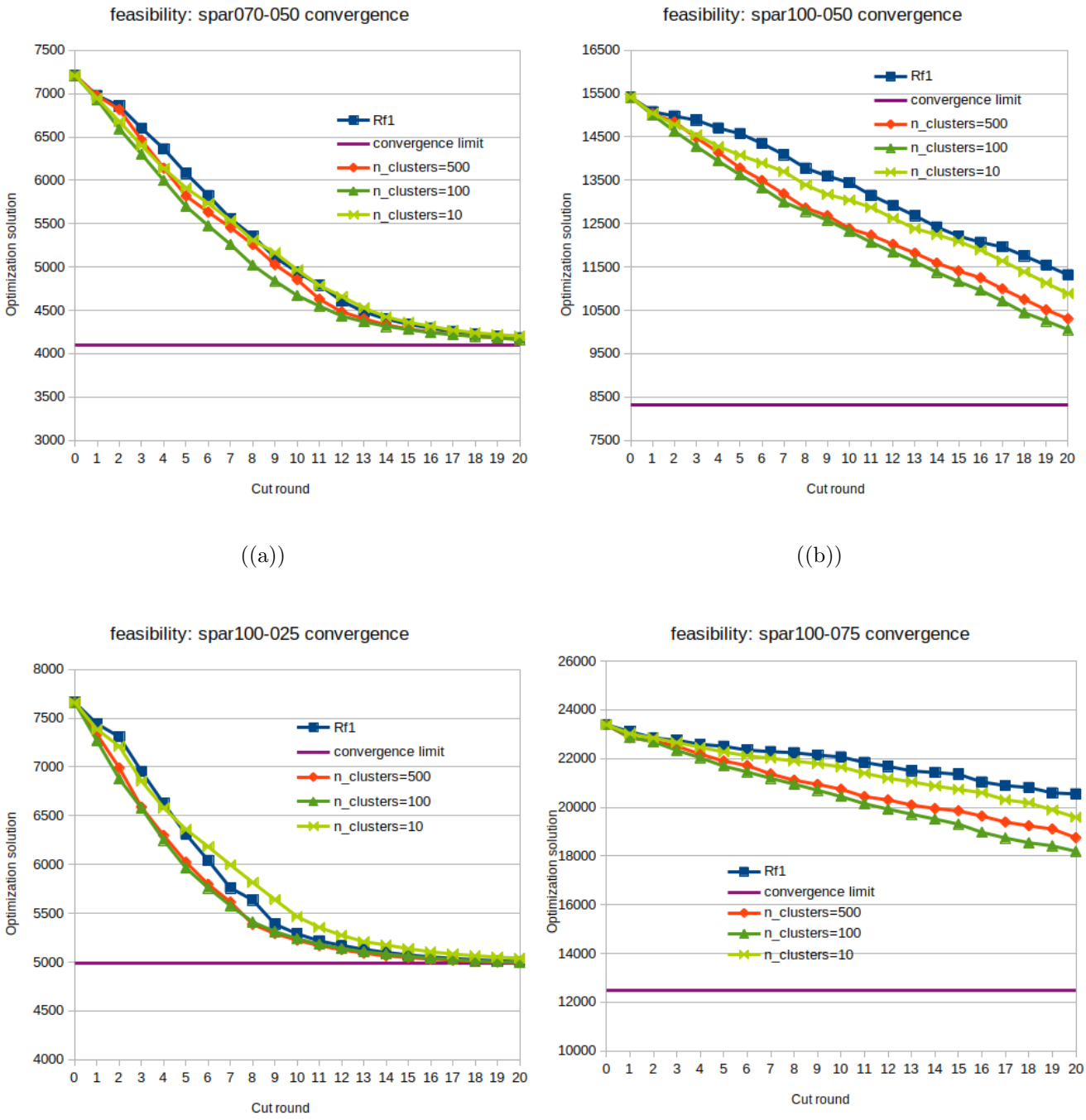


Figure 4.1: Comparison of dimensionality and sparsity for k-means

In the comparison of dimensionality the results reveal:

- Notable improvement (15%, difference with Rf1,of gap closure of k-means(k=100) with *convergence limit*) in the case of high dimensional problem, Fig.4.1b. k-means algorithm results in lower final bounds and faster convergence over Rf1. To compare the overall gap closure with the computational experiments of 3.1 we would need to change the selection measure from 100 cuts/round to 5% of ρ sub-problems.

- Marginal improvement in the case of medium dimensional problem, Fig.4.1a. Rf1 is providing a significant gap closure with *convergence limit*, thus there is a small room for improvement for k-means algorithm. Even though

For the comparison of sparsity we observe:

- Notable and almost consistent improvement (18%, difference with Rf1,of gap closure with *convergence limit* for k=100) of k-means in comparison to Rf1 in the case of high density problem, Fig.4.1d

- Improved performance of k-means for all different cluster numbers, Fig.4.1d. The gap between Rf1 and *convergence limit* is the greatest among the set of experiments. The proposed k-means algorithm is bridging this gap, however the gap closure of k-means with *convergence limit* still remains significant.

- When the number of clusters is equal to the number of added cuts/round the best performance is observed. In order to verify the dependence of number of clusters with the number of cuts/round additional computational experiments are to be considered for varying # of clusters=# of added cuts/round.

- Marginal improvement in the case of low density problem, with k=100 having the best performance, Fig.4.1c. k-means for k=10 results in inferior convergence and lower bounds over Rf1.

Overall k-means is proven to be a useful tool for cut selection, especially in the cases where the reference algorithm is struggling the most, high dimensional and high density problems. k-means for k=100 results in a final solution that is 12% reduced over Rf1, for *spar100-075*, in real world applications this could result in a significant total cost reduction for example.

Figure 4.2 presents the results for combined selection measure (100 variables and 50% density) as opposed to Fig.4.1b which presents the results for feasibility selection. In Fig.4.1b k-means algorithm indicates an improved convergence over Rf1. In contrast, k-means for combined selection has inferior performance to the reference algorithm Rf2 for any # of clusters. In addition, while in all studied cases with feasibility measure (Fig.4.1) the optimization solution for k-means follows the same trajectory as Rf1. Here Rf2 is having a curved trajectory and converges faster and in a lower bound than Rf1, yet k-means can not take advantage of this potential. Comparing k-means for feasibility in Fig.4.1b with Rf2 Fig.4.2 , Rf2 provides a lower bound and faster convergence indicating that the proposed k-means approach can not replace the combined selection with the trained neural network [2].

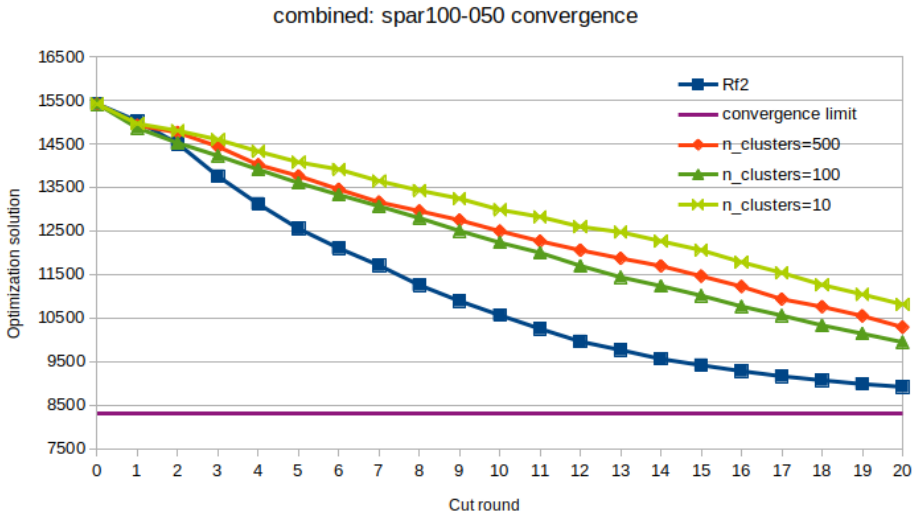


Figure 4.2: Combined selection measure for k-means

4.3 Agglomerative clustering

Hierarchical clustering is a general family of clustering algorithms that build nested clusters by merging or splitting them successively. This hierarchy of clusters is represented as a tree (or dendrogram). The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample.

Agglomerative clustering is a bottom-up approach of hierarchical clustering. Each observation starts as a cluster and clusters are successively merged together. The linkage criteria determines the metric used for the merge strategy:

- **Ward** minimizes the sum of squared differences within all clusters. It is a variance-minimizing approach and in this sense is similar to the k-means objective function but tackled with an agglomerative hierarchical approach.
- **Maximum or complete linkage**minimizes the maximum distance between observations

of pairs of clusters.

- **Average linkage** minimizes the average of the distances between all observations of pairs of clusters.

- **Single linkage**minimizes the distance between the closest observations of pairs of clusters.

Agglomerative clustering algorithm is derived from the scikit-learn python package [21] defining the number of clusters setting linkage to ward and thus affinity to Euclidean, the rest parameters are set to default.

4.4 Round of Experiments 2: Agglomerative clustering - Results

In the second round of experiments Algorithm 2 with Agglomerative clustering is deployed. The parameters are the final number of clusters (k=10,100). The results of Round of Experiment 2 are juxtaposed with the results of Round of Experiments 1 in Figure 4.3 . Agglomerative clustering indicates similar behavior with k-means for k=100 and a slightly better performance in the case of k=10. Both of-the self clustering approaches result in the improvement of the reference algorithm Rf1.

By using the ward linkage we where expecting to see similar results to k-means since the clusters are evaluated in order to achieve the minimum variance. The rationale behind the Agglomerative clustering is that all data points are of equal importance in the initialization of the algorithm while k-means initialization is based on the distribution of the data set. The results

indicate that the clustering approach, hierarchical or k-means, is not affecting the optimization algorithm. Varying the linkage for Agglomerative clustering can be examined in the future, to see the full potential of this approach.

Agglomerative clustering can be implemented using different norms,e.g. taxicab distance, manhattan distance and cosine similarity. The alternative norms for computing the distance between the clustering approach could result in an improved performance, however there are no indications by the original data set to follow these geometries. However a norm that is exploiting the geometry of the data set, could provide significant improvements over the Euclidean norm. This new norm will be introduced in the following chapter.

Observing that there are no deviations between Agglomerative clustering and k-means for Euclidean norm the combined selection computational experiments were omitted, because the results are expected be analogous to Fig.4.2.

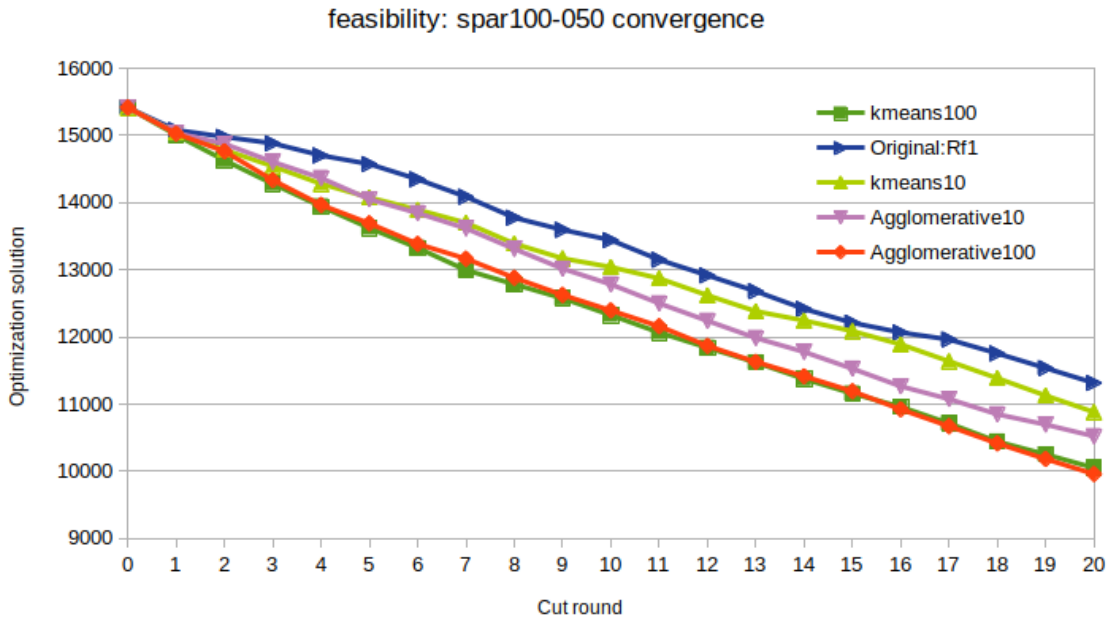


Figure 4.3: Comparison of k-means and Agglomerative clustering in Algorithm 2

Chapter 5

A new space distance for the *separation problem*: the affinity norm

The results for clustering using the Euclidean norm where encouraging, indicating that there can be improvements in the Master problem given an extra sorting of the selected cutting planes. By introducing the tailor-made norm/affinity norm we examine the potential of the customization of the clustering approach based on better understanding the *separation problem*. While in k-means and Agglomerative clustering the proximity of the $\tilde{x_\rho}$ was evaluated by the Euclidean norm $\|N\|_{af}$ and not taking into consideration the ranking proposed by Algorithm 8, the affinity norm evaluates $\tilde{x_\rho}$ based on their ranking by Algorithm 8, feasibility or combined.

Initially the first element of the ranking($\tilde{x_\rho}$), corresponding to the sub-problem with the most negative eigenvalue (if feasibility measure is selected), is considered to be the centroid of the cluster, the rest of the elements are clustered with the centroid iff they share at least two same variables with the centroid. $\|N\|_{af}$ compounds differences in the type of active variables at each sub-set. Different criteria have been implemented in order to further evaluate the cor-

relation of the clustered elements. Based on discrete criterion $C1$, the elements within a cluster are all considered overlapping. Criterion $C2$ examines the sub-clusters within a cluster and dictates which sub-clusters contain overlapping elements either with a discrete approach$C2_a$ or a continuous approach $C2_b$.The rationale in the affinity norm has been to assist in spreading out the sub-sets over the full space rather than stow and congest candidates in adjacent locations.

In addition a hybrid algorithm combining of-the-self and tailor-made clustering approaches has also been examined to examine the synergy of the $\|N\|_{af}$ with SoA clustering algorithms.

5.1 Affinity norm

Let $S := \{1, 2, \cdots N\}$ be a totally ordered set. We introduce the symmetric group S_n containing all ordered permutations of S with $n \leq N$ elements.

Let y' be the reference permutation in $S_m, m \geq n$ and y any permutation in S_n.

We introduce the affinity norm such that,

$$\|N\|_{af} := \sum_{i=1}^{n} g(y_i),$$

where

$$g(y_i) = \begin{cases} 1, \text{ if } \exists\ j \mid y_i = y'_j \\ \\ 0, \text{ otherwise} \end{cases}$$

If $\|N\|_{af} = 2$ then y shares the same 2 active variables as y'. We create a cluster initiated by y' containing all vectors for which $\|N\|_{af} = 2$ is satisfied.

For the purpose of this study we set $n = m = 3$.

To further extend this procedure we group the vectors satisfying $\|N\|_{af} = 2$ into 3 sub-clusters based on which active variables they share with y'.

To merge the $\|N\|_{af}$ with the original optimization problem we consider each permutation as an n slice of the original variable x corresponding to a ρ sub-problem, $y \sim x_\rho$ ans $y' \sim x'_\rho$

Depending on how we interpret the results of the tailor-made clustering we have examined the following criteria as to decide upon the representative elements of each cluster.

1. **Discrete** $C1$

   Based on this criterion all elements clustered with x'_ρ are considered overlapping and only x'_ρ holds valuable information about the original space, and thus is considered the representative element of the cluster.

2. **Discrete** $C2_a$

   Based on $C2_a$ not all elements clustered with x'_ρ are considered overlapping. It is a more conservative approach as to $C1$, given that we consider overlapping only the elements in the sub-cluster which contains the most elements.

3. **Continuous** $C2_b$

   After the tailor-made clustering is completed the standard deviation STD of $\tilde{x_\rho}$ in each sub-cluster can be computed. We consider informative all the vectors contained in the sub-cluster with the maximum STD.

The methodology of the use of affinity norm is explained in detail in Algorithm 3.

Algorithm 3: Iterative SDP outer-approx. with cut selection/generation based on Affinity norm

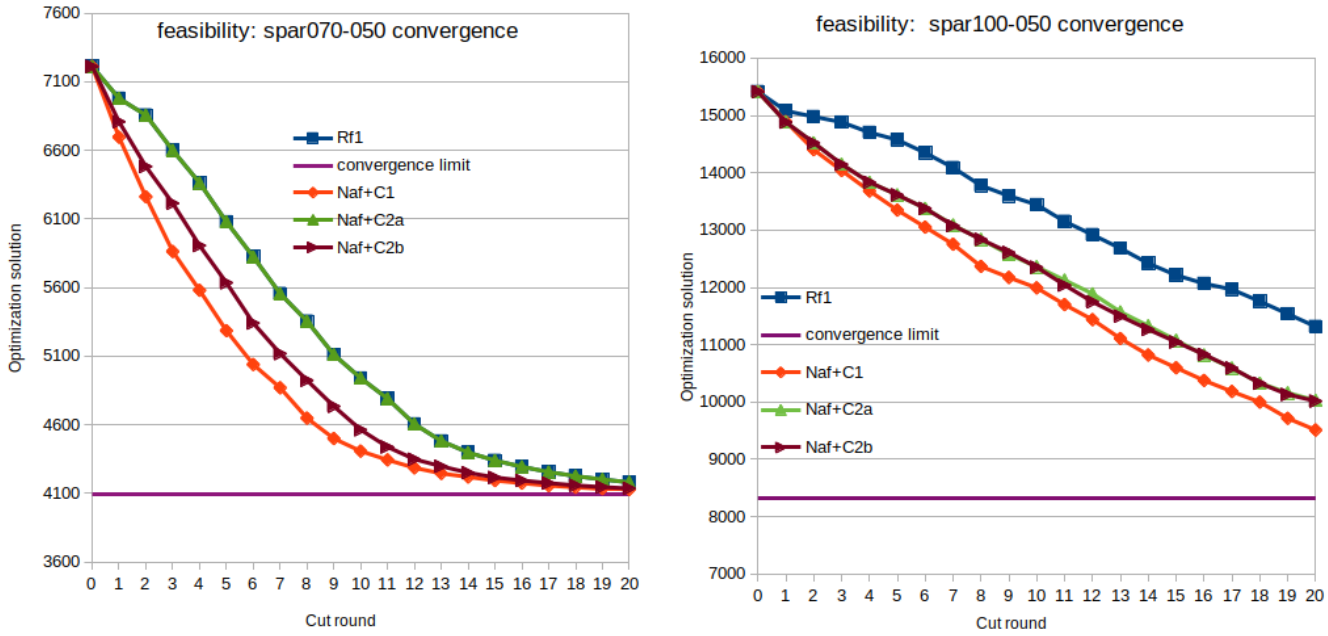
input :
-current base LP relaxation of $\mathcal{B}$ of QP, either fully added from the start,i.e. $\mathcal{M}$ or separates iteratively at each cut round;
-decomposed SDP relax. $\mathcal{S}(\mathcal{F})$ to outer-approx., where $\mathcal{B} \subseteq \mathcal{P}_n$ with small n;
-incumbent LP solution $(\tilde{x}, \tilde{X})$;
- selection strategy/ordering metric $M(\rho)\forall\rho \in \mathcal{F}$ at $(\tilde{x}, \tilde{X})$ e.g. $-\lambda_{min}(\rho), \mathcal{C}(\rho)$ etc.;
-selection size, i.e. a fixed % of $|\mathcal{F}|$or a fixed number of cuts (set to 100);
-number of cut rounds $\mathcal{R}$ (set to 20);
-termination criteria, if active terminate on an improvement between to consecutive cut rounds of$\leq$ 0.01% of the gap closed overall so far from the $\mathcal{M}$ bound;
- criterion for sorting clustered elements e.g. C1, C2a and C2b;
-maximum number of points $\mathcal{MN}$ to be examined with Affinity norm for a reference point x' (set to 1000)

output : Polyhedral outer-approximation that lower bounds $z(\mathcal{B}+\mathcal{S}(\mathcal{F}))$ and SDP relax.$z(\mathcal{B}+\mathcal{S})$;

for $\mathcal{R}$ *cut round if termination criteria not met* **do**
  Sort $\mathcal{F}$ by descending $M(\rho)\forall\rho \in \mathcal{F}$ at current$(\tilde{x}, \tilde{X})$;
  for *top* ρ *sub-problems in* $\mathcal{F}$ *within maximum number* $\mathcal{MN}$ **do**
    Fix $\tilde{x}_\rho$ as $\tilde{x}'$ being the initial element of a cluster $K = K \cap \tilde{x}'$;
    for *following top* ρ *sub-problems in sorted* $\mathcal{F}$ *within maximum number of* $\mathcal{MN}-1$ **do**
      Apply $\|N\|_{af}$;
      if $\|N\|_{af} = 2$ **then**
        Cluster $\tilde{x}_\rho$ with $\tilde{x}'$ in $K = K \cup \tilde{x}_\rho$
    Apply selection criterion to discard elements in K
    Renew $\mathcal{F}$ based on the discarded elements of K
  for *top* ρ *sub-problems in sorted* $\mathcal{F}$ *within selection size* **do**
    if $-\lambda_{min}(\rho) < 0$ *(viol. PSD condition for* $\begin{bmatrix} 1 & \tilde{x_\rho} \\ \tilde{x_\rho}^T & \tilde{X}_\rho \end{bmatrix}$*)* **then**
      $\mathcal{B} = \mathcal{B} \cup$ {new EigCut (ρ) based on $-\lambda_{min}(\rho)$}
  Resolve (warm-start) new LP relaxation $\mathcal{B}$ that includes added cuts;
  Update current incumbent solution $(\tilde{x}, \tilde{X})$;
Last obtained $z(\mathcal{B})$ lower bounds $z(\mathcal{B}+\mathcal{S}(\mathcal{F}))$ and $z(\mathcal{B}+\mathcal{S})$;

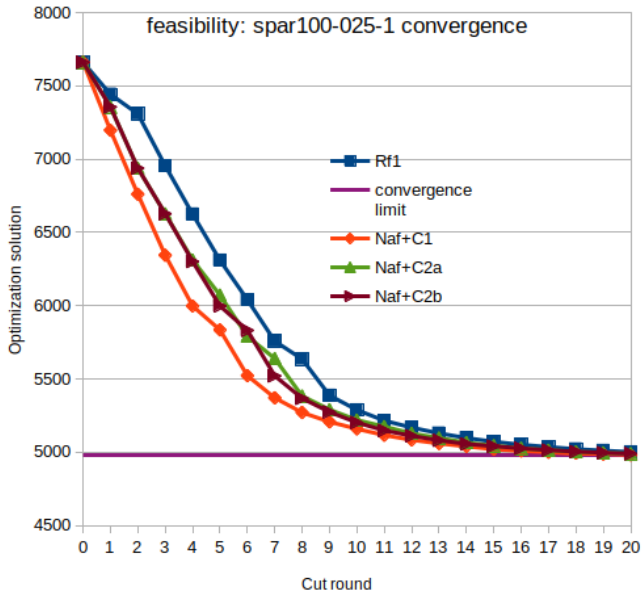
5.2 Round of Experiments : affinity norm - Results

The potential of Algorithm 3 is examined in Round of Experiments 3, the parameters are the criterion to evaluate the created clusters (C1,C2a, C2b), dimensinality, sparsity and selection measure (feasibility, combined). Figure 5.1 illustrates the results for different dimensionality : Fig.5.1a with 70 variables and Fig.5.1b with 100 variables, and for different density: Fig.5.1b for 100 variables and 50% density, Fig.5.1c for 25% density and Fig.5.1d for 75% density.

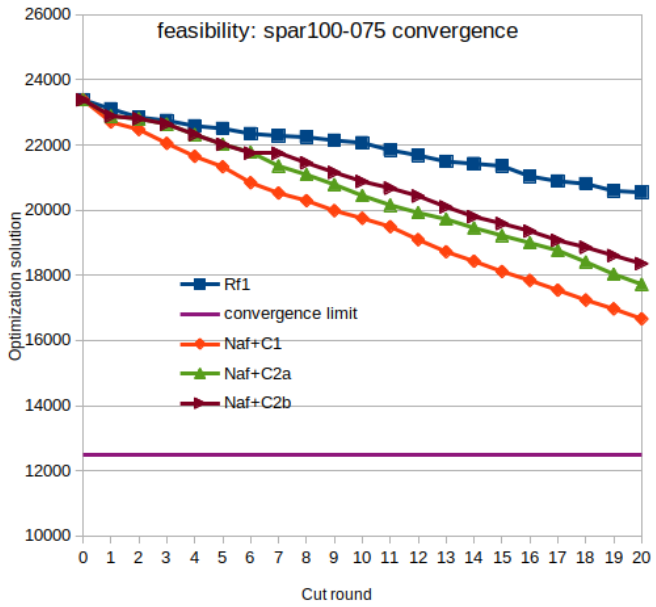


((a))

((b))



((c))



((d))

Figure 5.1: Comparison of dimensionality and sparsity for Affinity norm & selection criteria

In the comparison of dimensionality, Fig.5.1a and Fig.5.1b, we observe :

- Significant convergence improvement with $\|N\|_{af} + C_1$ (21%, difference with Rf1, of gap

closure with *convergence limit*) and lower bound in the case of high dimensional problem. $\|N\|_{af}$ and C2 result in improved performance as well.

- Marginal improvement of $\|N\|_{af}$ for medium dimensional problem. $\|N\|_{af}$ with criteria C1 and C2b are resulting in improved convergence. Seemingly $\|N\|_{af} + C2a$ and Rf1 are overlapping, yet there is a slight improvement by $\|N\|_{af} + C2a$

- $\|N\|_{af} + C_1$ has the best performance in both instances indicating that a brute approach discarding the majority of the cutting planes selected by the $\|N\|_{af}$ is the most favorable.

In the comparison of sparsity there is:

- Significant and consistent improvement by $\|N\|_{af}$ (30%,difference with Rf1, of gap closure with *convergence limit*) for $\|N\|_{af} + C1$ even in high density problems

- Marginal and consistent improvement for small density problem

- For the high dimensional problem and different sparsities Fig.5.1b, Fig.5.1c and Fig.5.1d, criteria $C2a$ and $C2b$ are having a similar and consistent impact on the optimization algorithm. The idea of creating sub-clusters within the created clusters by $\|N\|_{af}$ has not proved to be significantly useful under the proposed selection criteria.

In all cases $\|N\|_{af}$ is resulting in consistent improvement over Rf1. In the case of, *spar100-075* the final solution provided by $\|N\|_{af} + C1$ is 19% reduced over the solution provided from Rf1, and 8% reduced over k-means for k=100.

In Figure 5.2 the results of Algorithm 3 for affinity norm and combined selection measure are illustrated. There is a marginal improvement by $\|N\|_{af}$ for all selection measures. The $\|N\|_{af}$ is taking advantage of the improvement of Rf2 over Rf1 and provides an even better convergence.

Comparing the convergence of $\|N\|_{af} + C1$ in Fig.5.1b with Rf2 in Fig.5.2 indicates that the proposed affinity norm can compete the introduced neural network [2] in terms of convergence and final bounds. In the case of *spar100-050* the solution of Rf2 is reduced by 22% compared to Rf1, while the solution of $\|N\|_{af} + C1$ for feasibility selection is reduced by 16% compared to Rf1.

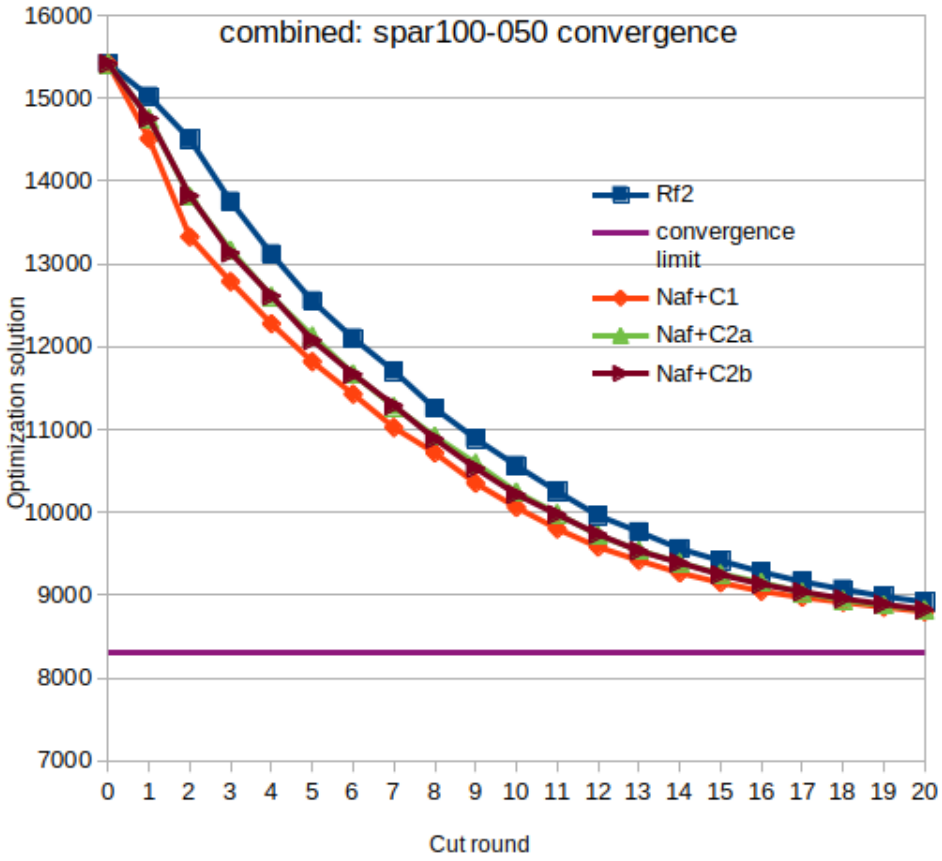


Figure 5.2: Combined selection measure for Affinity norm & selection criteria

5.3 Hybrid clustering

We further deploy a hybrid clustering approach combining the of-the-self clustering algorithms with the tailor-made affinity norm. The two algorithms are used sequentially and thus there

are two variations of the hybrid approach.

- **Hybrid 1**

  In this variation, k-means algorithm is used first followed by the implementation of affinity norm $\|N_{af}\|$. K-means creates the initial clusters and affinity norm is used to sort the elements within the cluster. Hybrid 1 introduces a ranking of the clusters created by k-means and the number of cuts created from each cluster depends on that ranking. The ranking of clusters is performed based on maximum selection measure $M(\rho)$ of their containing sub-problems. The aim of this addition is to examine if by selecting cuts in specific areas instead of the entire space could give better results.

Algorithm 4: Iterative SDP outer-approx. with cut selection/generation based on Hybrid 1 clustering

input :
-current base LP relaxation of $\mathcal{B}$ of QP, either fully added from the start,i.e. $\mathcal{M}$ or separates iteratively at each cut round;
-decomposed SDP relax. $\mathcal{S}(\mathcal{F})$ to outer-approx., where $\mathcal{B} \subseteq \mathcal{P}_n$ with small n;
-incumbent LP solution $(\tilde{x}, \tilde{X})$;
- selection strategy/ordering metric $M(\rho) \forall \rho \in \mathcal{F}$ at $(\tilde{x}, \tilde{X})$ e.g. $-\lambda_{min}(\rho), \mathcal{C}(\rho)$ etc.;
-selection size, i.e. a fixed % of $|\mathcal{F}|$or a fixed number of cuts (set to 100);
-number of cut rounds $\mathcal{R}$ (set to 20);
-termination criteria, if active terminate on an improvement between to consecutive cut rounds of$\leq$ 0.01% of the gap closed overall so far from the $\mathcal{M}$ bound;
-of-the-self clustering technique (k-means or Agglomerative clustering);
- number of clusters k ;
- criterion for sorting clustered elements e.g. C1, C2a and C2b;
-maximum number of points $\mathcal{MN}$ to be examined with Affinity norm for a reference point x' (set to 1000);
-number of cuts created from the top ranked cluster N_C

output : Polyhedral outer-approximation that lower bounds $z(\mathcal{B} + \mathcal{S}(\mathcal{F}))$ and SDP relax.$z(\mathcal{B} + \mathcal{S})$;

for $\mathcal{R}$ *cut round if termination criteria not met* **do**
  Cluster all elements in $\mathcal{F}$ (off-the self clustering);
  for *Every cluster* **do**
    Sort all elements in cluster by descending $M(\rho) \forall \rho \in \mathcal{F}$ at current$(\tilde{x}, \tilde{X})$;
    for *top ρ sub-problems in cluster within maximum number* $\mathcal{MN}$ **do**
      Fix $\tilde{x}_\rho$ as $\tilde{x}'$ being the initial element of a cluster $K = K \cap \tilde{x}'$;
      for *following top ρ sub-problems in sorted $\mathcal{F}$ within maximum number of* $\mathcal{MN} - 1$ **do**
        Apply $\|N\|_{af}$;
        if $\|N\|_{af} = 2$ **then**
          Cluster $\tilde{x}_\rho$ with $\tilde{x}'$ in $K = K \cup \tilde{x}_\rho$
      Apply selection criterion to discard elements in K
      Renew clusters elements based on the discarded elements of K
    if $k \leq$ *selection size* **then**
      for *top selection size/k sub-problems in cluster* **do**
        $\mathcal{B} = \mathcal{B} \cup \{$new EigCut (ρ) based on $-\lambda_{min}(\rho)$ $\}$
  if $k \geq$ *selection size* **then**
    Sort clusters based on the sub-problem with maximum $M(\rho)$ they contain;
    for *cluster in sorted clusters* **do**
      for *top ρ sub-problems within cluster and within* N_C **do**
        $\mathcal{B} = \mathcal{B} \cup \{$ new EigCut (ρ)based on $-\lambda_{min}(\rho)\}$
      Redefine $N_C = g(N_C)$ to compute the created cuts at the following cluster
  Resolve (warm-start) new LP relaxation $\mathcal{B}$ that includes added cuts;
  Update current incumbent solution $(\tilde{x}, \tilde{X})$;
Last obtained $z(\mathcal{B})$ lower bounds $z(\mathcal{B} + \mathcal{S}(\mathcal{F}))$ and $z(\mathcal{B} + \mathcal{S})$;

Hybrid 2

In Hybrid 2 algorithm the affinity norm is applied first followed by k-means clustering. The affinity norm is used to sort the initial population so that k-means will reveal underlying correlations. Algorithm 5 represent the Hybrid 2 algorithm.

Algorithm 5: Iterative SDP outer-approx. with cut selection/generation based on Hybrid 2 clustering

input :
-current base LP relaxation of $\mathcal{B}$ of QP, either fully added from the start,i.e. $\mathcal{M}$ or separates iteratively at each cut round;
-decomposed SDP relax. $\mathcal{S}(\mathcal{F})$ to outer-approx., where $\mathcal{B} \subseteq \mathcal{P}_n$ with small n;
-incumbent LP solution $(\tilde{x}, \tilde{X})$;
- selection strategy/ordering metric $M(\rho) \forall \rho \in \mathcal{F}$ at $(\tilde{x}, \tilde{X})$ e.g. $-\lambda_{min}(\rho), \mathcal{C}(\rho)$ etc.;
-selection size, i.e. a fixed % of $|\mathcal{F}|$or a fixed number of cuts (set to 100);
-number of cut rounds $\mathcal{R}$ (set to 20);
-termination criteria, if active terminate on an improvement between to consecutive cut rounds of$\leq$ 0.01% of the gap closed overall so far from the $\mathcal{M}$ bound;
-of-the-self clustering technique (k-means or Agglomerative clustering);
- number of clusters k ;
- criterion for sorting clustered elements e.g. C1, C2a and C2b;
-maximum number of points $\mathcal{MN}$ to be examined with Affinity norm for a reference point x' (set to 1000)

output : Polyhedral outer-approximation that lower bounds $z(\mathcal{B} + \mathcal{S}(\mathcal{F}))$ and SDP relax.$z(\mathcal{B} + \mathcal{S})$;

for $\mathcal{R}$ *cut round if termination criteria not met* **do**
  Sort $\mathcal{F}$ by descending $M(\rho) \forall \rho \in \mathcal{F}$ at current$(\tilde{x}, \tilde{X})$;
  for *top* ρ *sub-problems in* $\mathcal{F}$ *within maximum number* $\mathcal{MN}$ **do**
    Fix $\tilde{x}_\rho$ as $\tilde{x}'$ being the initial element of a cluster $K = K \cap \tilde{x}'$;
    for *following top* ρ *sub-problems in sorted*$\mathcal{F}$ *within maximum number of* $\mathcal{MN} - 1$ **do**
      Apply $\|N\|_{af}$;
      if $\|N\|_{af} = 2$ **then**
        Cluster $\tilde{x}_\rho$ with $\tilde{x}'$ in $K = K \cup \tilde{x}_\rho$
    Apply selection criterion to discard elements in K
    Renew $\mathcal{F}$ based on the discarded elements of K
  Cluster all elements in $\mathcal{F}$ (off-the self clustering);
  for *Every cluster* **do**
    Sort all elements in cluster by descending $M(\rho) \forall \rho \in \mathcal{F}$ at current$(\tilde{x}, \tilde{X})$;
    if $k \geq$ *selection size* **then**
      Create EigCut (ρ) based on $-\lambda_{min}(\rho)$ for the top (1st)sorted element in cluster;
      Let the set Eg containing all the selected eigencuts then $Eg = Eg \cup \{EigCut(\rho)\}$;
    if $k \leq$ *selection size* **then**
      for *top selection size/k sub-problems in cluster* **do**
        Create EigCut (ρ) based on $-\lambda_{min}(\rho)$ and $Eg = Eg \cup \{EigCut(\rho)\}$
  Sort EigCut (ρ) in Eg based on $M(\rho)$;
  for *top EigCut* (ρ) *in Eg within selection size* **do**
    $\mathcal{B} = \mathcal{B} \cup \{$ EigCut $(\rho)\}$
  Resolve (warm-start) new LP relaxation $\mathcal{B}$ that includes added cuts;
  Update current incumbent solution $(\tilde{x}, \tilde{X})$
Last obtained $z(\mathcal{B})$ lower bounds $z(\mathcal{B} + \mathcal{S}(\mathcal{F}))$ and $z(\mathcal{B} + \mathcal{S})$;

5.4 Round of Experiments 4: Hybrid - Results

In the fourth and final round of experiments the Algorithm 4, Hybrid 1 clustering, and Algorithm 5, Hybrid 2, are examined for different dimensionality and sparsity of the original problems.

Figure 5.3 illustrated the gap closed between *convergence limit* and all of the examined algorithms for varying dimensionality and sparsity.

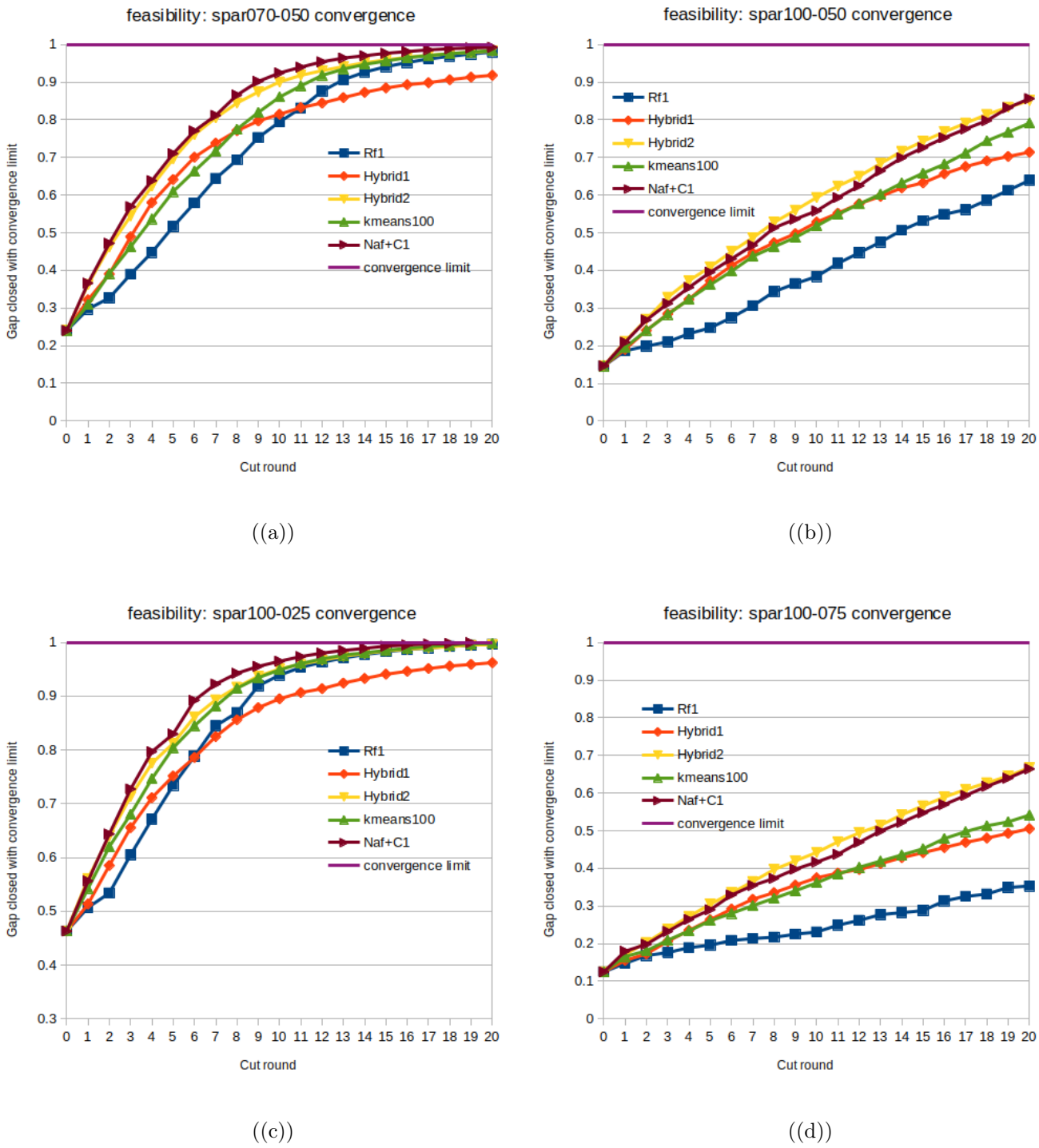


Figure 5.3: Comparison of dimensionality sparsity for the proposed clustering approaches

In the comparison of dimensionality we observe:

- Improvement of Hybrid algorithms compared to Rf1
- Marginal improvement of Hybrid 2 compared to $\|N\|_{af}$
- Hybrid 2 is resulting in a greater gap(91% gap closed) with *convergence limit* than Rf1(98% gap closed) in the medium dimensionality problem Fig.6a

In the comparison of sparsity, the results reveal:

- Consistent similar convergence of Hybrid 2 and $\|N\|_{af}$ such as Hybrid 1 and k-means in the cases of high and medium density problems, Fig.6b and Fig.6d, suggest the dominance of the initial step of the hybrid algorithms over the convergence of optimization
- Hybrid 2 is resulting in a greater gap(96% gap closed) with *convergence limit* than Rf1(99% gap closed) in the low density problem
- When the gap between Rf1 and *convergence limit* is significant, the proposed algorithms yield more favorable results. In the cases of small to medium dimensionality (20-90 variables) regardless of density, and in high dimensional problems with small density, the proposed clustering approaches provide marginal improvements

Figure 5.4 and Figure 5.5 present the results concerning the final solution, reached in 20 rounds, and the required computational time, using Intel® Core™ i7-4510U CPU @ 2.00GHz×4.

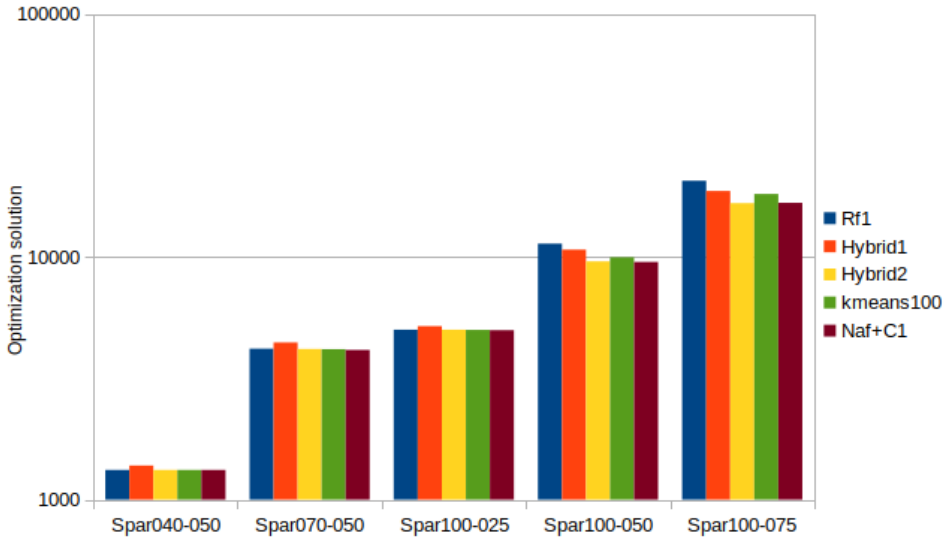


Figure 5.4: Final solution reached in 20 rounds: feasibility measure and clustering approaches

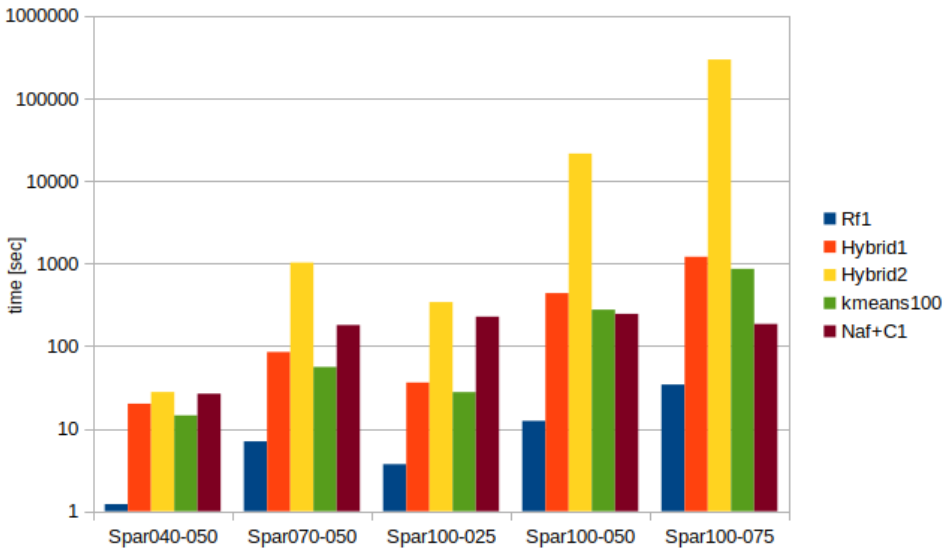


Figure 5.5: Computational time for 20 rounds: feasibility measure and clustering approaches

Overall the clustering approaches improve the final solution in the case of high dimensional and high density problems. The tailor-made $\|N\|_{af}$ provides the best results (minimum final solution). In terms of computational time Hybrid 2 has significant delays over the rest clustering approaches and Rf1. The $\|N\|_{af} + C1$ small and medium dimensionality (50% density) and high dimensionality (25%) density is delaying over Hybrid1 and kmeans. However in high

dimensional problem high density $\|N\|_{af} + C1$ is converging faster(in CPU time), this is due to the introduced maximum number of elements $\mathcal{MN}$ in Algorithm 3 dictating that not all of the available cutting planes should be examined since only 100 cuts are added in the optimization problem in each round. In addition to time acceleration, the use of $\mathcal{MN}$ provides an other insight to the problem, suggesting that the ranking of the sub-problems by the original Algorithm 8 is of high importance and should not be overlooked.

The computational time for the optimization algorithm using k-means clustering is increasing with the complexity of the problem , increased dimensionality and density, this is due to the fact that the population of the data set is increasing as well.

Chapter 6

Conclusions and further research

Based on the desiderata of the computational experiments we can conclude that :

- There is indeed a correlation in the selected cutting planes with previous algorithm (Rf)
- New norm has contributed to **significant and consistent improvement** of all previous variations of the algorithm
- Of-the-self norms provide improvements of lesser significance, erratic and occasionally inferior to the nominal algorithm
- Hybridized, the norm retained a potential but without significant signs of improvement
- Clustering approach has a greater impact in the case of high dimensional & medium- high density problems

Future research should address the issues of computational delays of the affinity norm in order to accelerate the proposed methodology. Implementing the affinity norm iteratively over the ranked sub-problems by feasibility measure could be replaced by a more efficient algorithm.

In addition,the extension of the algorithms in order to exploit the temporal data and discover similarity patters across iterations should be considered. The aim of the present research was to examine the correlation of the selected cutting planes within the same cut round. A temporal research should examine the repetition of the selected cutting planes over iterations. The repetition can be examined by two perspectives, in one case the repeating variables of the selected cutting planes could be slowing down the optimization algorithm if incumbent solution is away from the global optimum, in the other case repeating variables could indicate the convergence around the global optimum. A more flexible approach can be examined promoting that in the initial cut rounds cutting planes with high variance are desirable in order to span the variable space, and as the optimization algorithm converges overlapping cutting planes are favored in order to tighten the relaxation around the global minimum.

A wider range of BoxQP can be examined ,e.g.QPlib [13] or different type of QP such as quadratically constrained quadratic programs (QCQP), in order to examine the potential of the introduced affinity norm in problems that SoA solvers are having difficulties to address. Apart from varying the QP problems, different SoA MINLP solvers, e.g. Mosek, BARON, ANTIGONE, should be examined as well to explore the potential of the proposed clustering methods.

The affinity norm was introduced for cutting plane selection in outer-approximating QP. Without requiring any reformulations, affinity norm can be implemented in evaluating the population of low-dimensional cutting planes available for any type of optimization problems e.g MILP, MINLP.

Finally we have examined the case of 3-D cutting planes, all of the clustering approaches can be adjusted for different dimensionality of cutting planes.

Appendices

Appendix A

Final bound and computational time

A.1 % final bound closure between different cut selection approaches and *convergence limit* in 20 rounds

instance	Rf1	kmeans100	$\|N\|_{af} + C1$	Hybrid1	Hybrid2
spar040-050	100.00	100.00	99.99	95.74	99.99
Spar070-050	97.95	98.37	99.22	91.80	98.22
Spar100-025	97.95	99.82	100.05	96.22	99.61
Spar100-050	63.93	71.40	85.11	79.11	85.63
Spar100-075	35.28	54.15	66.41	50.53	66.76

A.2 % difference of final bound between Rf1 and clustering approaches in 20 rounds

instance	kmeans100	$\|N\|_{af} + C1$	Hybrid1	Hybrid2
Spar040-050	0.00	0.00	4.26	0.01
Spar070-050	-0.41	-1.25	6.02	-0.27
Spar100-025	-0.20	-0.39	3.42	0.04
Spar100-050	-12.10	-15.95	-5.49	-15.43
Spar100-075	-11.79	-18.89	-9.26	-19.11

A.3 CPU time(sec) for clustering approaches and Rf1 in 20 rounds

instance	Rf1	kmeans100	$\|N\|_{af} + C1$	Hybrid1	Hybrid2
Spar040-050	1.22	20.00	27.71	14.48	26.49
Spar070-050	7.01	84.70	1025.66	55.69	179.60
Spar100-025	3.71	36.12	339.91	27.67	226.90
Spar100-050	12.40	436.89	21447.94	275.15	245.35
Spar100-075	34.17	1208.31	294805.48	860.25	184.87

Appendix B

Results of computational experiments

B.1 Round of Experiments 1

Cut round	Rf1	k=500	k=100	k=10	*convergence limit*
0	7210.75	7210.75	7210.75	7210.75	4097.77
1	6977.97	6976.41	6929.84	6946.58	4097.77
2	6856.28	6816.78	6592.37	6678.06	4097.77
3	6599.63	6463.82	6300.36	6400.40	4097.77
4	6362.26	6139.55	5997.24	6135.67	4097.77
5	6078.78	5819.06	5698.55	5912.16	4097.77
6	5823.97	5632.11	5475.76	5733.10	4097.77
7	5556.07	5455.22	5259.21	5535.85	4097.77
8	5355.08	5258.98	5019.86	5309.57	4097.77
9	5110.96	5026.36	4837.76	5164.71	4097.77
10	4940.53	4852.39	4670.23	4966.37	4097.77
11	4791.03	4627.54	4548.93	4786.27	4097.77
12	4606.23	4482.87	4434.31	4658.86	4097.77
13	4481.90	4402.23	4366.45	4528.37	4097.77
14	4398.99	4331.84	4314.96	4423.25	4097.77
15	4340.29	4283.26	4277.89	4360.99	4097.77
16	4294.84	4250.21	4242.01	4316.61	4097.77
17	4256.60	4222.34	4219.03	4269.76	4097.77
18	4227.00	4200.67	4199.16	4241.60	4097.77
19	4202.48	4181.98	4181.20	4220.37	4097.77
20	4181.87	4163.40	4164.56	4202.58	4097.77

Table B.1: Spar070-050: results of final bounds for k-means and feasibility selection

Cut round	Rf1	k=500	k=100	k=10
0	7660.75	7660.75	7660.75	7660.75
1	7442.03	7333.23	7266.49	7384.63
2	7308.77	6989.85	6878.94	7212.64
3	6954.34	6587.99	6577.92	6855.50
4	6624.54	6297.75	6249.08	6579.88
5	6309.91	6023.37	5963.00	6359.02
6	6040.64	5796.58	5759.27	6182.25
7	5758.71	5614.09	5577.27	5996.09
8	5635.51	5386.38	5410.22	5815.08
9	5387.84	5291.28	5312.43	5640.17
10	5288.95	5224.01	5239.81	5465.35
11	5216.21	5167.67	5181.35	5354.29
12	5169.06	5126.57	5137.84	5273.67
13	5130.16	5090.57	5104.16	5208.14
14	5098.00	5063.30	5081.24	5174.08
15	5072.50	5045.90	5059.62	5136.06
16	5052.06	5029.14	5039.30	5104.11
17	5036.77	5017.49	5024.77	5081.32
18	5022.15	5007.04	5011.12	5064.81
19	5012.01	4999.59	5003.18	5050.62
20	5002.60	4993.23	4994.35	5038.70

Table B.2: Spar100-025: results of final bounds for k-means and feasibility selection

Cut round	Rf1	k=500	k=100	k=10
0	15415.75	15415.75	15415.75	15415.75
1	15081.65	15035.83	15006.29	15041.87
2	14979.68	14844.67	14631.39	14787.48
3	14884.03	14480.62	14280.35	14542.02
4	14704.07	14144.99	13944.18	14280.80
5	14575.16	13781.41	13623.58	14077.41
6	14348.66	13496.21	13326.05	13896.32
7	14088.83	13184.66	12997.00	13701.28
8	13776.95	12857.97	12785.68	13391.03
9	13597.03	12682.52	12577.88	13171.80
10	13443.18	12386.25	12325.49	13042.10
11	13151.17	12233.28	12067.47	12873.93
12	12918.35	12021.74	11841.14	12621.75
13	12681.91	11819.59	11621.81	12384.42
14	12419.33	11590.62	11376.44	12245.36
15	12212.77	11406.82	11159.27	12088.16
16	12067.98	11248.50	10960.93	11890.67
17	11964.32	10994.36	10714.84	11639.10
18	11756.22	10750.87	10444.70	11386.96
19	11538.03	10510.91	10253.66	11129.09
20	11316.75	10304.21	10052.83	10883.26

Table B.3: Spar100-050: results of final bounds for k-means and feasibility selection

Cut round	Rf1	k=500	k=100	k=10
0	23387.50	23387.50	23387.50	23387.50
1	23107.85	22847.41	22880.77	23020.57
2	22847.55	22724.86	22688.83	22845.05
3	22750.46	22510.99	22338.17	22662.14
4	22585.55	22181.46	22031.79	22463.19
5	22504.74	21898.44	21692.25	22273.51
6	22347.31	21713.49	21450.72	22114.94
7	22283.33	21366.96	21190.89	22011.46
8	22241.88	21112.13	20950.80	21901.11
9	22140.68	20941.92	20701.91	21794.14
10	22067.83	20741.63	20439.04	21655.68
11	21841.09	20437.48	20146.47	21400.22
12	21682.13	20303.21	19923.28	21191.04
13	21491.96	20093.68	19718.37	21046.79
14	21426.72	19948.83	19514.87	20875.74
15	21356.63	19855.58	19309.20	20737.05
16	21039.17	19637.99	18973.64	20599.22
17	20889.98	19394.80	18741.57	20304.00
18	20809.98	19242.16	18547.34	20180.86
19	20592.02	19107.97	18416.98	19887.56
20	20546.51	18750.62	18192.70	19587.31

Table B.4: Spar100-075: results of final bounds for k-means and feasibility selection

Cut round	Rf2	k=500	k=100	k=10	*convergence limit*
0	15415.75	15415.75	15415.75	15415.75	8316.57
1	15022.34	14922.47	14862.54	14961.85	8316.57
2	14504.57	14761.21	14521.33	14801.72	8316.57
3	13755.78	14434.97	14224.51	14596.88	8316.57
4	13120.75	14019.14	13910.70	14325.90	8316.57
5	12552.75	13761.89	13602.85	14078.61	8316.57
6	12103.77	13456.12	13329.95	13913.24	8316.57
7	11707.61	13160.72	13068.14	13645.87	8316.57
8	11255.63	12958.55	12797.36	13429.38	8316.57
9	10891.06	12748.34	12503.27	13243.87	8316.57
10	10565.05	12496.94	12232.29	12986.36	8316.57
11	10252.41	12264.32	12000.39	12824.19	8316.57
12	9961.52	12058.26	11699.78	12596.86	8316.57
13	9766.39	11871.23	11438.49	12474.33	8316.57
14	9562.02	11695.94	11235.00	12265.41	8316.57
15	9418.09	11462.66	11015.80	12059.13	8316.57
16	9281.67	11223.81	10767.57	11783.19	8316.57
17	9164.75	10933.93	10558.48	11538.63	8316.57
18	9072.30	10759.29	10332.64	11263.05	8316.57
19	8985.87	10549.63	10142.60	11044.02	8316.57
20	8920.53	10291.23	9946.34	10812.80	8316.57

Table B.5: Spar100-050: results of final bounds for k-means and combined selection

B.2 Round of experiments 2

Cut round	k=500	k=100	k=10
0	15415.75	15415.75	15415.75
1	15054.92	15028.93	15038.42
2	14896.46	14765.39	14876.91
3	14657.64	14335.63	14607.16
4	14294.18	13967.27	14367.28
5	13920.20	13691.86	14051.64
6	13657.61	13386.34	13845.62
7	13344.07	13166.58	13617.15
8	13073.73	12883.03	13312.10
9	12803.36	12626.70	13018.71
10	12537.86	12395.72	12784.72
11	12292.65	12160.46	12502.11
12	12018.59	11868.93	12241.08
13	11782.68	11629.31	11985.81
14	11606.50	11415.92	11778.63
15	11404.31	11192.77	11528.64
16	11196.77	10923.92	11267.00
17	10992.78	10672.72	11077.53
18	10763.58	10416.78	10850.69
19	10541.85	10184.94	10697.08
20	10326.42	9961.00	10523.46

Table B.6: Spar100-050: results of final bounds for Agglomerative clustering and feasibility selection

B.3 Round of experiments 3

Cut round	$\|N\|_{af} + C1$	$\|N\|_{af} + C2a$	$\|N\|_{af} + C2b$
0	7210.75	7210.75	7210.75
1	6695.94	6977.97	6808.25
2	6261.77	6856.28	6481.93
3	5863.20	6599.63	6214.25
4	5579.96	6362.26	5906.26
5	5285.95	6078.78	5635.32
6	5039.58	5823.97	5341.10
7	4870.04	5556.07	5119.10
8	4648.86	5355.08	4924.40
9	4500.15	5110.96	4734.91
10	4408.54	4940.53	4564.17
11	4344.91	4791.03	4439.02
12	4287.02	4606.23	4349.65
13	4246.29	4481.90	4298.03
14	4220.49	4398.99	4251.79
15	4194.52	4340.29	4217.19
16	4175.06	4294.84	4195.15
17	4157.16	4256.60	4176.00
18	4144.78	4227.00	4159.30
19	4135.79	4202.48	4147.03
20	4129.59	4181.87	4137.39

Table B.7: Spar070-050: results of final bounds for affinity norm and feasibility selection

Cut round	$\|N\|_{af} + C1$	$\|N\|_{af} + C2a$	$\|N\|_{af} + C2b$
0	7660.75	7660.75	7660.75
1	7196.37	7355.64	7355.64
2	6760.89	6938.55	6938.55
3	6344.32	6627.57	6627.57
4	5998.35	6311.99	6301.71
5	5834.66	6072.48	5998.69
6	5522.50	5791.40	5831.95
7	5371.36	5639.83	5521.47
8	5271.59	5384.04	5368.28
9	5207.89	5291.02	5274.90
10	5160.17	5220.57	5203.25
11	5116.80	5176.07	5149.44
12	5084.73	5129.11	5112.30
13	5060.16	5097.53	5081.73
14	5040.59	5066.84	5057.20
15	5020.43	5045.22	5041.97
16	5009.57	5027.70	5027.62
17	5000.57	5014.50	5014.95
18	4994.51	5003.98	5005.08
19	4987.85	4997.13	4997.00
20	4982.87	4990.46	4990.58

Table B.8: Spar100-025: results of final bounds for affinity norm and feasibility selection

Cut round	$\|N\|_{af} + C1$	$\|N\|_{af} + C2a$	$\|N\|_{af} + C2b$
0	15415.75	15415.75	15415.75
1	14893.18	14884.56	14884.56
2	14402.48	14524.68	14524.68
3	14040.02	14147.93	14147.93
4	13682.69	13834.61	13834.61
5	13347.80	13614.44	13614.44
6	13051.66	13379.48	13379.48
7	12752.42	13088.45	13078.87
8	12366.20	12837.69	12838.07
9	12173.37	12576.56	12609.73
10	11991.53	12360.27	12352.06
11	11699.70	12126.65	12039.84
12	11437.08	11890.20	11748.82
13	11108.38	11572.48	11497.85
14	10820.26	11329.00	11266.51
15	10596.90	11076.72	11047.46
16	10376.41	10820.23	10830.08
17	10183.11	10591.77	10594.83
18	9997.80	10327.94	10327.09
19	9715.59	10161.03	10131.11
20	9511.89	10027.75	10016.45

Table B.9: Spar100-050: results of final bounds for affinity norm and feasibility selection

Cut round	$\|N\|_{af} + C1$	$\|N\|_{af} + C2a$	$\|N\|_{af} + C2b$
0	23387.50	23387.50	23387.50
1	22703.73	22889.22	22889.22
2	22475.00	22808.95	22808.95
3	22049.76	22641.74	22641.74
4	21651.25	22321.64	22321.64
5	21330.68	22020.21	22020.21
6	20842.34	21779.13	21750.84
7	20526.00	21356.72	21746.91
8	20291.61	21089.46	21453.34
9	19984.87	20785.58	21158.94
10	19750.75	20444.69	20881.43
11	19493.06	20157.31	20677.81
12	19095.75	19920.35	20434.71
13	18722.82	19722.86	20101.45
14	18432.75	19454.13	19799.89
15	18117.38	19220.95	19590.85
16	17844.59	18994.51	19357.91
17	17543.99	18762.72	19073.72
18	17241.72	18410.38	18861.15
19	16970.70	18034.30	18607.29
20	16664.29	17723.81	18361.03

Table B.10: Spar100-075: results of final bounds for affinity norm and feasibility selection

Cut round	$\|N\|_{af} + C1$	$\|N\|_{af} + C2a$	$\|N\|_{af} + C2b$
0	15415.75	15415.75	15415.75
1	14510.16	14756.99	14756.99
2	13326.24	13813.67	13823.91
3	12785.78	13166.13	13132.55
4	12277.67	12609.10	12614.25
5	11819.00	12130.36	12078.98
6	11427.83	11673.78	11670.19
7	11025.14	11277.67	11292.21
8	10717.50	10920.43	10886.69
9	10352.62	10593.18	10534.08
10	10060.63	10247.50	10217.52
11	9800.66	9985.00	9974.99
12	9582.20	9725.66	9735.57
13	9419.37	9540.88	9534.76
14	9269.66	9394.34	9392.41
15	9151.91	9259.00	9248.87
16	9048.42	9154.35	9133.05
17	8974.81	9034.92	9042.30
18	8916.46	8943.72	8959.71
19	8855.71	8880.50	8893.06
20	8800.60	8826.51	8826.48

Table B.11: Spar100-050: results of final bounds for affinity norm and combined selection

B.4 Round of Experiments 4

Cut round	Hybrid 1	Hybrid 2
0	7210.75	7210.75
1	6877.70	6709.07
2	6596.42	6302.42
3	6190.92	5964.34
4	5818.81	5639.02
5	5567.87	5344.14
6	5324.96	5079.39
7	5172.22	4896.65
8	5036.73	4735.06
9	4931.20	4614.70
10	4857.53	4506.53
11	4785.24	4434.31
12	4736.16	4383.18
13	4676.29	4338.71
14	4619.45	4296.46
15	4571.46	4266.04
16	4536.29	4237.25
17	4514.68	4217.26
18	4482.34	4199.06
19	4453.90	4184.03
20	4433.75	4170.51

Table B.12: Spar070-050: results of final bounds for Hybrid clustering and feasibility selection

Cut round	Hybrid 1	Hybrid 2
0	7660.75	7660.75
1	7410.52	7173.96
2	7053.32	6779.60
3	6701.78	6412.40
4	6424.81	6105.49
5	6224.42	5921.00
6	6051.49	5675.74
7	5858.07	5518.06
8	5702.67	5400.03
9	5590.29	5298.08
10	5508.17	5232.18
11	5450.43	5187.37
12	5414.40	5150.57
13	5362.48	5115.23
14	5319.74	5087.85
15	5279.84	5064.83
16	5253.57	5048.26
17	5227.77	5037.56
18	5204.71	5025.56
19	5189.26	5013.01
20	5173.88	5004.72

Table B.13: Spar100-025: results of final bounds for Hybrid clustering and feasibility selection

Cut round	Hybrid 1	Hybrid 2
0	15415.75	15415.75
1	15050.20	14873.15
2	14632.05	14387.20
3	14268.99	13899.32
4	13947.40	13533.41
5	13538.28	13230.05
6	13201.86	12893.73
7	12922.95	12585.55
8	12695.63	12245.34
9	12495.05	11977.29
10	12240.34	11696.29
11	12041.73	11443.90
12	11833.29	11227.19
13	11671.74	10946.09
14	11484.99	10676.54
15	11373.80	10454.45
16	11172.98	10249.72
17	11014.12	10053.68
18	10888.25	9863.37
19	10790.36	9704.32
20	10695.20	9554.82

Table B.14: Spar100-050: results of final bounds for Hybrid clustering and feasibility selection

Cut round	Hybrid 1	Hybrid 2
0	23387.50	23387.50
1	23008.22	22817.23
2	22792.58	22413.37
3	22377.57	21975.22
4	22013.24	21550.29
5	21659.04	21149.44
6	21307.55	20769.79
7	20982.14	20396.18
8	20764.82	19998.60
9	20515.28	19711.80
10	20272.81	19431.95
11	20115.67	19079.80
12	19996.49	18784.57
13	19803.69	18532.08
14	19605.58	18181.35
15	19440.03	17888.89
16	19271.74	17597.55
17	19097.76	17346.54
18	18956.81	17124.63
19	18799.89	16901.52
20	18644.58	16619.67

Table B.15: Spar100-075: results of final bounds for Hybrid clustering and feasibility selection

Bibliography

[1] K. M. Anstreicher, *Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming.* J. of Global Optimization, 43(2-3):471 – 484, 2009

[2] Radu Baltean-Lugojan, Ruth Misener, Pierre Bonami and Andrea Tramontani, *Selecting cutting planes for quadratic semidefinite outer-approximation via trained neural networks.* 2018.

[3] Mokhtar S. Bazaraa, Hanif D. Sherali, C. M. Shetty *Nonlinear Programming: Theory and Algorithms* John Wiley Sons, Inc., Hoboken, New Jersey, 2006

[4] E. A. Boyd, *Fenchel Cutting Planes for Integer Programs* Operations Research, 42(1): 2-196, 1994

[5] Boyd, Stephen and Vandenberghe, Lieven, *Convex Optimization* Cambridge University Press, USA,2004

[6] Stephen P. Bradley, Arnoldo C. Hax, Thomas L. Magnanti *Applied Mathematical Programming* Addison-Wesley Publishing Company, 1977

[7] M. Bramer *Principles of Data Mining* Springer-Verlag London Ltd, 2016

[8] S. Burer *Optimizing a polyhedral-semidefinite relaxation of completely positive programs.* Mathematical Programming Computation. 2(1), 119 ,2010

[9] S. Burer and D. Vandenbussche. *Globally solving box-constrained nonconvex quadratic programs with semidefinite-based finite branch-and-bound.* Comput. Optim. Appl., 43(2): 181-195, 2009

[10] M. A. Duran and I. E. Grossmann , *An outer-approximation algorithm for a class of mixed-integer nonlinear programs* J. Mathematical programming, 36(3): 307-339, 1986.

[11] C.A Floudas, *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*, Topics in Chemical Engineering, Oxford University Press,1995

[12] C.A. Floudas, C.E. Gounaris, *A review of recent advances in global optimization.* J. Glob Optim volume45, 3, 2009.

[13] F. Furini, E. Traversi, P. Belotti, A. Frangioni, A. Gleixner, N. Gould, L. Liberti, A. Lodi, R. Misener, H. Mittelmann, N. V. Sahinidis, S. Vigerske, and A. Wiegele. *QPLIB: a library of quadratic programming instances.* Mathematical Programming Computation, 11, 237–265,2019

[14] Jiyao Gao *Branch and Bound (BB)* Northwestern University Process Optimization Open Textbook, available on https://optimization.mccormick.northwestern.edu/index.php/Main$_P age$

[15] A. M. Geoffrion, Generalized benders decomposition. J. Optim. Theory and Appl.,10(4):237, 1972.

[16] R. E. Gomory, *Outline of an algorithm for integer solutions to linear programs.* Bull. American Math. Soc., 64:275, 1958.

[17] G. Guillén-Gosálbez, A. Sorribas *Identifying quantitative operation principles in metabolic pathways: a systematic method for searching feasible enzyme activity patterns leading to cellular adaptive responses.* BMC Bioinformatics 10, 386 (2009)

[18] David G. Luenberger *Optimization by Vector Space Methods* John Wiley Sons, Inc., New York, 1969

[19] J.E. Mitchell *Integer Programming: Branch and Cut Algorithms.* In: Floudas C., Pardalos P. (eds) Encyclopedia of Optimization. Springer, Boston, MA, 2008

[20] A. Neumaier, *Complete Search in Continuous Global Optimization and Constraint Satisfaction.* J. Acta Numerica, volume 13, 5, 2004.

[21] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E, *Scikit-learn: Machine learning in Python.* Journal of Machine Learning Research, volume 12, 2825–2830, 2011

[22] A. Qualizza, P. Belotti, and F. Margot, *Linear programming relaxations of quadratically constrained quadratic programs.* Mixed Integer Nonlinear Programming, volume 154 of The IMA Volumes in Mathematics and its Applications, pages 407–426. Springer New York, 2012.

[23] Shapiro, Alexander, and Andy Philpott. *A tutorial on stochastic programming.* Manuscript. Available at www2. isye. gatech. edu/ ashapiro/publications. html (2007)

[24] H. D. Sherali and B. M. P. Fraticelli, *Enhancing RLT Relaxations via a New Class of Semidefinite Cuts.* J. of Global Optimization, 22(1-4): 233-261. Kluwer Academic Publishers, 2002.

[25] D. Vandenbussche and G. Nemhauser. *A branch-and-cut algorithm for nonconvex quadratic programs with box constraints.* Mathematical Programming, 102(3):55-575, 2005.