Republic of Iraq

Ministry of Higher Education & Scientific Research

University of Babylon

College of Education for Pure Sciences

Department of Mathematics

Numerical Optimization Methods for Solving the Semidefinite Programming of NP-Hardness Problems

A Dissertation

Submitted to the Council of College of Education for Pure Sciences / University of Babylon in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Education / Mathematics.

By

Ahmed Hasan Hameed ALRIDHA

Supervised

Asst. Prof. Dr. Ahmed Sabah Al-Jilawi

2022 A.D. 1443 A.H.

بسم الله الرحمن الرحيم

وَلَقَدْ كَتَبْنَا فِي الزَّبُورِ مِنْ بَعْدِ الذِّكْرِ أَنَّ الْأَرْضَ يَرِثُهَا عِبَادِيَ الصَّالِحُونَ

صَدَقَ اللهُ العَلِيُّ العَظِيم

سورة الأنبياء
الآية (105)

Supervisor's Certification

I certify that the dissertation entitled **"Numerical Optimization Methods for Solving the Semidefinite Programming of NP-Hardness Problems"** by **"Ahmed Hasan Hameed ALRIDHA"** has been prepared under my supervision in Babylon University/ College of Education for Pure Sciences as a partial requirement for the Degree of Doctor of Philosophy in Education / Mathematics.

Signature:

Name: Dr. Ahmed Sabah Al-Jilawi

Title: Assistant Professor

Date: / / 2022

In view of the available recommendation, I forward this dissertation for debate by the examining committee.

Signature:

Name: Dr. Azal Jaafar Musa

Head of Mathematics Department

Title: Assistant Professor

Date: / / 2022

Certification of linguistic Expert

I certify that I have read this dissertation entitled " **"Numerical Optimization Methods for Solving the Semidefinite Programming of NP-Hardness Problems"** and corrected its grammatical mistakes; therefore, it has qualified for debate.

Signature:

Name: Dr. Waffa Mukhles Faisal

Title: Assistant Professor

Address: English Department, University of Babylon

Date: / / 2022

Certification of Scientific Expert

I certify that I have read the scientific content of this dissertation **"Numerical Optimization Methods for Solving the Semidefinite Programming of NP-Hardness Problems"** and I have approved this dissertation is qualified for debate.

Signature:

Name: Dr. Bassem Abbas Hassan

Title: Professor

Address: College of Computer Science and Mathematics, University of Mosul

Date: / / 2022

Certification of Scientific Expert

I certify that I have read the scientific content of this dissertation **"Numerical Optimization Methods for Solving the Semidefinite Programming of NP-Hardness Problems"** and I have approved this dissertation is qualified for debate.

Signature:

Name: Dr. Mushtaq Karim Abed Alrahim

Title: Assistant Professor

Address: College of Administration and Economics, University of Karbala

Date: / / 2022

Dedication

To the soul of my little girl (SANA).

Ahmed Hasan Hamed Mahmud.

Acknowledgements

In life there are wonderful phenomena that happen only once and may not be repeated, and this fact was embodied by the knowledge of Prof. Dr. Ahmed Sabah Al-Jilawi who was and still is a true brother and friend in addition to being a supervisor. I extend my sincere thanks and gratitude to his honorable person.

To all my teachers.

To my family.

To my loyal friends.

Abstract

The dissertation focus on NP-hard problem (non deterministic polynomial time) with large scale variable. Whereas, the K-CLUSTER PROBLEM (search for a densest subgraph of fixed size k) with semidefinite bounds which is a classical NP-hard problem in combinatorial optimization and has been demonstrated to be an NP-Hardness problem in graph clustering. The strategy was generates a good bound to the original problem's optimum value. The augmented Lagrangian method has been developed as a replacement for the penalty method. As well as, a hybrid technique that changes between the two ways has been developed based on the value of the parameter. The new model was tested using many graphs from the Biq Mac library, where the Julia language was applied to obtain the results. A novel approach outperforms the penalty methods in terms of speed and robustness. Finally, the theoretical convergence properties of the proposed algorithm were demonstrating that it is the technique of choice for solving the semidefinite relaxations of binary quadratic problems.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table 1: **Symbols List**

$f(x)$	Objective Function
$Minf(x)$	The Minimum of Objective Function
$Maxf(x)$	The Maximum of Objective Function
$g(x)$	The Inequality Constraint
$h(x)$	The Equality Constraint
C	The Convex Set
$\mathbb{R}$	Real Numbers
$\nabla f(x)$	The Gradient of f
$\nabla^2 f(x)$	The Hessian of f
x^*	The Minimizer (Local, Global)
x^{k+1}	The Next Iterate
x^k	The Current Iterate
α_k	The Step Length

Table 2: **Abbreviations List**

$DOPs$	Discrete Optimization Problems
LP	Linear Programming Problems
TSP	Traveling Salesman Problem
MST	Minimum Spanning Tree
QP	Quadratic Programming
SDP	Semidefinite Programming
PEN	Penalty
$fcalls$	function calls
AUG	Augmented
$NLPP$	Non-Linear Programming Problems
KKT	Karush-Kuhn-Tucker Conditions
$\mathcal{C}$	The Cone
$\mathcal{L}$	The Lagrangian Function
NP	Non Deterministic Polynomial Time
GP	General Problem

CHAPTER 1

INTRODUCTION

1.1 Introduction

The general goal of problems with optimization is to achieve a solution with optimum objective function values [91]. In fact, optimization means finding the best solution (maximum or minimum) to the problem according to the objective function. Besides, being an important branch of mathematics, optimization is a fundamental approach that is commonly applied in our public life and in various fields. A combinatorial problem consists of finding one that satisfies a set of constraints among a finite set of objects [129]. Combinatorial optimization is a branch of mathematics optimization that involves selecting the best object from a finite collection of options, where the set of viable options is discrete or may be reduced to a discrete set. Many combinatorial optimization problems are NP hard problems such that the Traveling Salesman Problem ("TSP"), the Minimum Spanning Tree ("MST") problem, and the K-CLUSTER PROPLEM. Exhaustive search is not tractable in many of these problems, hence specialized algorithms that quickly rule out substantial areas of the search space or approximation techniques must be used instead. The optimization of many technically important combinatorial problems is the main goal. Optimization problems can be seen as generalizations of decision problems, where the solutions are additionally evaluated by an objective function and the goal is to find solutions with optimal objective function values [22]. The objective function is often defined on candidate solutions as well as on solutions; the objective function value of a given candidate solution (or solution) is also called its solution quality. Depending on whether the given objective function is to be minimized or maximized, any combinatorial optimization problem can be stated as a minimization problem or as a maximization question [122]. Sometimes, one of the two formulations is more natural, but problems of minimization and maximization are viewed equivalently algorithmically [28]. Furthermore, many algorithms for decision problems can be extended to related optimization problems in a rather natural way. However, such simple extensions of algorithms that work well on certain decision problems are not always effective for finding optimal or near-optimal

solutions of the corresponding optimization problems, and consequently, different algorithmic methods need to be considered for this task [63]. On the other hand, in the evolutionary computing community, numerical optimization problems are very common; all major evolutionary techniques as (genetic algorithm, evolutionary strategies, evolutionary programming) on these problems are very popular [36]. However, many of these tools seek to solve some real-world problems that involve not simple limitations (as well as other traditional optimization methods). The global solution also lies within the limits of the feasible region for problems like these. It is therefore important to investigate some of the problem-specific factors that effectively explore these limits [88]. Basically, two main types of problems can be classified (P and NP problems) where class P (problems) consists of all solvable decision problems in polynomial-time while NP (Hard problem) can be classified as follows:

1. The class of problems that cannot be solved in polynomial time is NP (that is hard to be solved) .

2. If its answer can not be verified in polynomial-time, the decision problem belongs to class NP hard.

In a general sense of computation, non-deterministic polynomial computation, that is NP represent the class of all non-deterministic polynomials. Thus, NP is the brief name of "Non deterministic Polynomial-time. The focus of our work will be on K-CLUSTER PROBLEM which is one of NP-hardness problem [113]. This problem generally binary quadratic problem and often difficult to be solved with normal application. The general form of optimization problems is

$$(P)\begin{cases}\max(\text{or } \min) & f_(x)\\ \text{subject to} & x \in X.\end{cases} \tag{1.1}$$

where $x \in R^n$ is a decision variable, $f(x)$ an objective function, $X \subset R^n$ a constraint set or feasible region [117]. Particularly, if the constraint set $X = R^n$, this optimization

problem is called an unconstrained optimization problem (assuming that the problem is to find minimize):

$$\min_{x \in R^n} f(x)$$

the constrained optimization problem can be written as follows:

$$\begin{aligned} \min_{x \in R^n} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) = 0, i \in E, \\ & h_i(x) \geq 0, i \in I, \end{aligned}$$

where E and I are the index set of constraints, $g_i(x), (i = 1, \cdots, m \in E \cup I)$ are constraint functions. When both objective function and constraint functions are linear functions, the problem is called linear programming [32]. Otherwise, the problem is called nonlinear programming. The optimization problem can be defined by the following flow chart (see Figure 1.1):

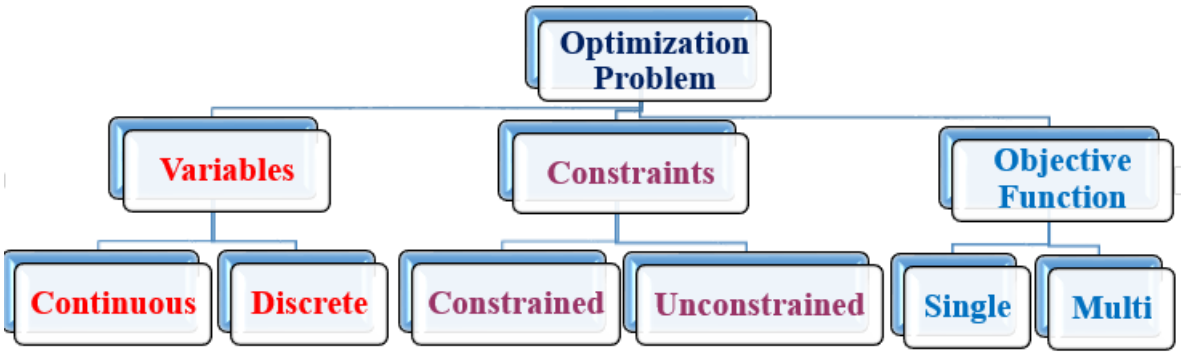

Figure 1.1: Flow Chart For the General of Optimization Problem.

As a result, the user's choice of important parameters in the optimization problem requires more precision. However, understanding the efficiency and speed of optimization methods,

which is primarily dependent on the number of specified design factors, is critical [36]. Finally, the optimization structure can be summarized by the following flow chart (see Figure (1.2)):

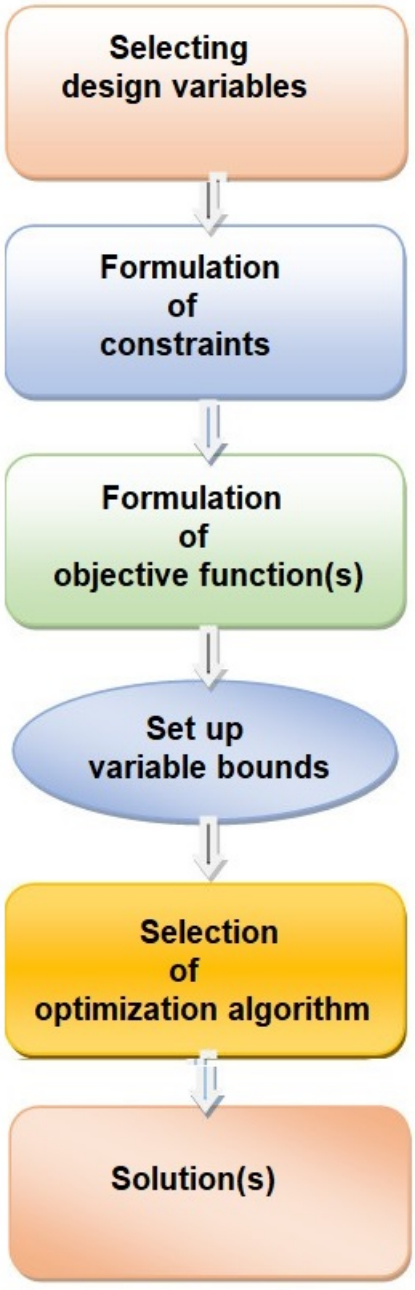

Figure 1.2: Flow Chart of The Optimal Design Method.

1.2 Related Work

Our research work is based on seven main sections, which will all be supportive of each other to achieve the goal:

1. Optimization Algorithms.
2. Discrete Optimization.
3. Clustering of Combinatorial Optimization.
4. NP-Hard in Cluster .
5. Semidefinite Program (SDP) in Combinatorial Optimization.
6. K-CLUSTER PROBLEM.
7. Approximate Method (Augmented Lagrangian and Penalty methods).

We endeavor to present all the above-mentioned sections in more detail

1.2.1 Optimization Algorithms

Since the 1970s, many optimization algorithms have been developed and applied to different optimization problems. Iterative optimization methods are used by starting with a guess for the variable x and iterate through a series of better guesses (called "iterates") until they reach a solution. The method for progressing from one iteration to the next separates the algorithm from others [105]. The values of the objective function of optimization problems, the constraint functions, and perhaps the first and second derivatives of these functions are used in most methods. Some algorithms save data from past iterations, while others rely solely on local data collected at the current moment [73]. Regardless of these details, the effective algorithms should have the following characteristics:

1. **Robustness :** An algorithms should able to perform well on a wide range of problems in their class, for any plausible beginning point values.

2. **Efficiency :** An algorithms shouldn't take up a lot of computer time or space.

3. **Accuracy :** An algorithms should be able to pinpoint a solution with pinpoint accuracy, without being excessively sensitive to data mistakes or arithmetic rounding errors that arise when the method is run on a computer.

A quickly converging approach for a large unconstrained nonlinear problem may need excessive computer storage. On the other hand, a sturdy approach may be the slowest. Numerical optimization is concerned with tradeoffs such as those between convergence rate and storage needs, robustness and speed, and so on [12]. In the following is a simplified diagram of the structure of optimization algorithm ,(see Figure 1.3):

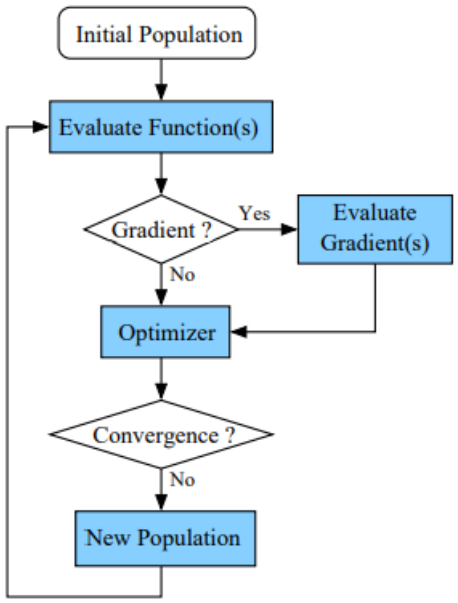

Figure 1.3: Flow chart of optimization algorithm

The mathematical theory of optimization is used both to characterize optimal points and to provide the basis for most algorithms [5]. It is not possible to have a good understanding of numerical optimization without a firm grasp of the supporting theory. The convergence

analysis methodology plays a major role in identifying the strengths and weaknesses of some of the most important algorithms. It is worth mentioning the convergence rate of an algorithm measures how quickly it approaches the best solution [63]. Local analysis, which focuses on the behavior of the algorithm in the vicinity of an optimal solution, is a traditional approach.

1.2.2 Discrete Optimization

The optimization in general deals with variables of a multitude of numeric data types, discrete optimization forces the variables to be strictly discrete and therefore limits the data types to be integer or binary. Discrete optimization problems are often encountered in reality and contain many of the famous problems for which efficient algorithms that can solve them in polynomial time are unknown, therefore said to be NP-hard to express their non-polynomial complexity [16]. Furthermore, Discrete Optimization Problems (DOPs) arise in various applications such as planning, scheduling, computer aided design, robotics and game playing. Often, a DOP is formulated in terms of finding a minimum cost solution path in a graph from an initial node to a goal node [34]. A discrete optimization problem can be given a declarative or procedural formulation, and both have their advantages. A declarative formulation simply states the constraints and objective function [121]. It allows one to describe what sort of solution one seeks without the distraction of algorithmic details [53]. A procedural formulation specifies how to search for a solution, and it therefore allows one to take advantage of insight into the problem in order to direct the search [53]. The ideal, of course, would be to have the best of all options, and this is the goal of constraint programming. According to this concept, it can be review a simplified example which is the 8-Puzzle problem. The eight Puzzle is made up of a 3×3-square grid with eight tiles numbered one through eight. One of the grid segments (referred to as the "blank") is blank. A tile can be transferred from an adjacent location into the blank position, resulting in a blank in the tile's original place. The aim is to go from a

starting position to a finishing position in the fewest amount of steps possible [55]. Looking at the following figure, the idea can be more clear (see Figure 1.4):

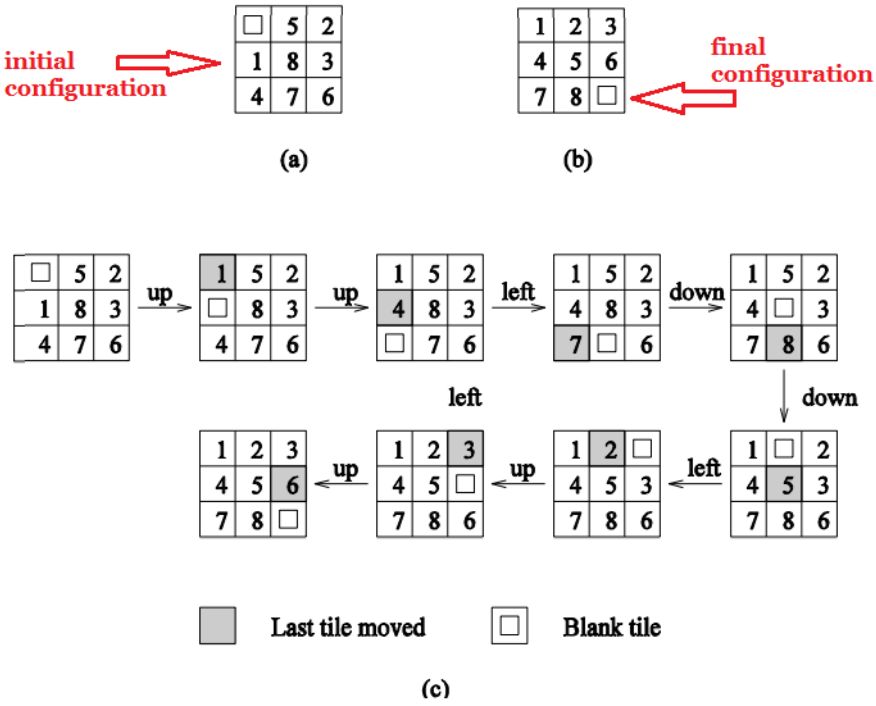

Figure 1.4: An example of an 8-Puzzle problem: (a) Initial configuration, (b) Final configuration, and (c) A sequence of moves going from the initial to the final configuration, with the cost equal to the number of steps in the sequence.

1.2.3 Clustering of Combinatorial Optimization

Many important combinatorial optimization problems can be cast as problems in NP, usually it would be dealing with approach the task of graph clustering from the common strategy of combinatorial optimization. In other words, the problem is formalized by introducing an objective function that assigns a numerical score to each discrete clustering of a graph, measuring how well it embodies community structure [119]. Optimizing the objective function over all possible clustering will then return "the best" partitioning of the graph with respect to a well-defined measure. As an example, one way to partition a graph G into two pieces is to identify a set of nodes S that minimizes the following

function f, referred to as the normalized cut objective [111, 120]:

$$f(S) = \frac{\text{number of edges leaving } S}{\text{number of edge endpoints in } S} + \frac{\text{number of edges leaving } S}{\text{number of edge endpoints not in } S}$$

Minimizing f will produce two clusters (nodes in S, and nodes not in S) that are both nontrivial in size and share few edges with each other [72].

1.2.4 NP-Hard in Clusters

When each connected component of G is a complete graph a graph G=(V, E) is called a cluster graph. Foe example, a complete subgraph of a graph (V, E) is called a clique, as in Figure (1.5) [25]. The decision problem is to find out if the graph has a clique of size k and the optimization problem is to find the max clique in the graph. A clique decision problem is NP hard problem for large graphs according to the following proof: If we are given a clique, we can easily determine if it is a clique of size k by counting the number of vertices in the clique; normally this answer can not be verified in polynomial-time for a large graphs, hence determining the size of a clique is an NP problem as in Figure (1.6), [33, 104].

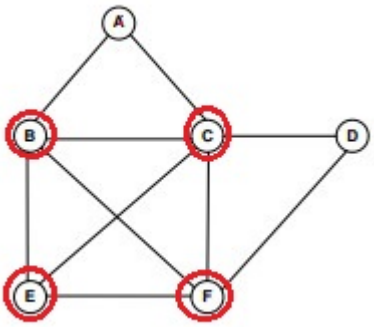

Figure 1.5: An Example of Cliques in a Graph.

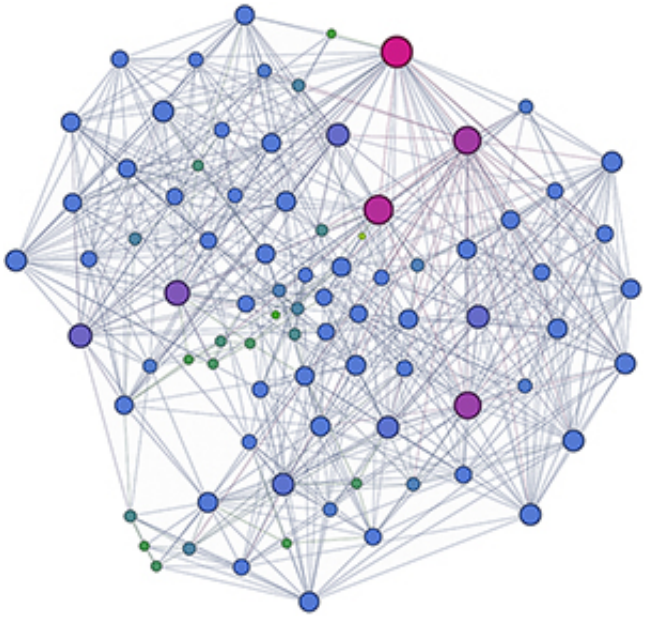

Figure 1.6: An Example of Max Clique in a Graph.

Finally, according to the following theory, the general framework is more concise :

Theorem 1.2.1. *In general, the problem of graph clustering is NP-hard.*

Proof: see [107].

1.2.5 Semidefinite Program (SDP) in Combinatorial Optimization

The semidefinite programming in combinatorial optimization is subfield of optimization. Actually, in recent years semidefinite programming has become a widely used tool for designing more efficient algorithms for approximating hard combinatorial optimization problems. Also, semidefinite programming, as opposed to linear programming, allows to work with positive semidefinite matrix variables. This field has recently generated a lot of interest, such as applications in optimization theory and control theory as well as the creation of effective internal point methods. However, semidefinite programming is not a novel concept in combinatorial optimization. Since the late 1960s, eigenvalue bounds have been given for combinatorial optimization problems. Semidefinite programs

may typically be recast as eigenvalue bounds. This reformulation is beneficial because it takes use of convex programming traits like duality and polynomial-time solvability while avoiding eigenvalue optimization problems. As shown in the figure below, the semidefinite program (SDP) field and its importance in that it includes several basic fields, the first of which is the linear programming field. For more see [44, 54, 75, 96, 114].

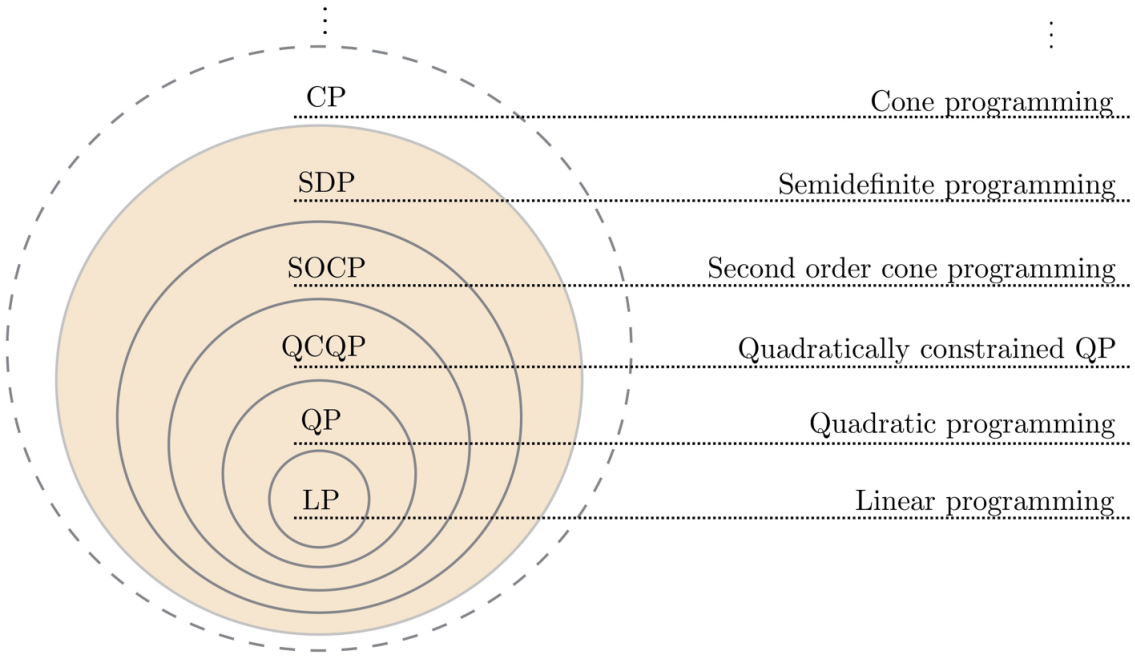

Figure 1.7: (SDP) and its position at the top of the convex optimization hierarchy.

1.2.6 K-CLUSTER PROBLEM

The K-CLUSTER problem entails finding a subgraph with the heaviest weight and exactly k nodes ($1 < k < n - 1$). In fact, this is a classical combinatorial optimization problem and has been shown to be NP-hard and also hard to approximate in (Deogun et al., 1997), [64, 80, 82, 90] . Previously, the K-CLUSTER problem was solved to optimality by the semidefinite-based branch-and-bound algorithm, which compared favorably with the convex quadratic relaxation method. Generally, clustering is a common multivariate data analysis procedure. It is intended to investigate the data objects' inherent natural structure, in which objects in the same cluster are as similar as possible and objects in

different clusters are as dissimilar as possible. The clusters' equivalence classes provide a way to generalize over the data objects and their attributes [3]. Clustering methods are used in a variety of fields, including medicine, psychology, economics, and pattern recognition. The term clustering is often confused with a classification or a discriminant analysis. The two types of data analysis refer to distinct concepts and are distinguished as follows:

1. Clustering differs from classification in that, the classification allocates items to pre-defined classes, whereas clustering requires no previous knowledge of the object classes or their members.

2. The cluster analysis differs from a discriminant analysis in that the former seeks to enhance an existing classification by reinforcing class demarcations, whilst the latter requires first establishing the class structure. (see Figures 1.8 and 1.9).

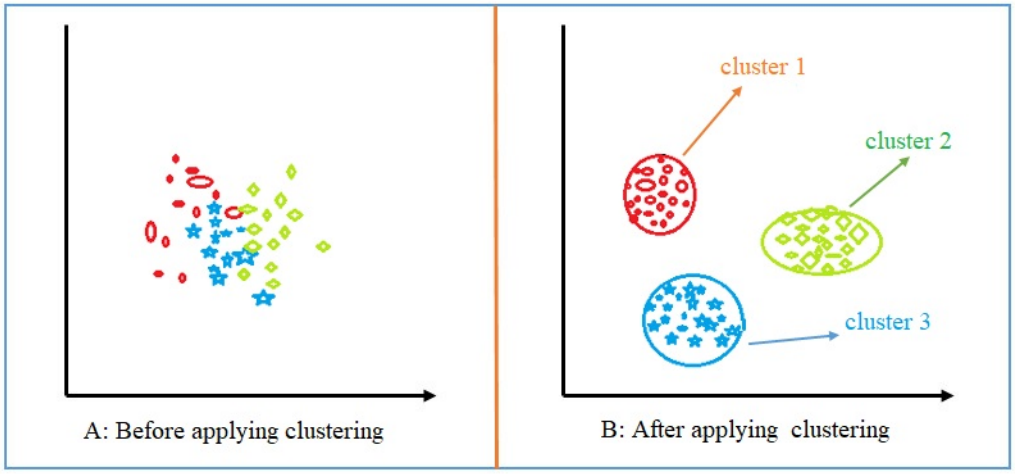

Figure 1.8: An example of cluster analysis.

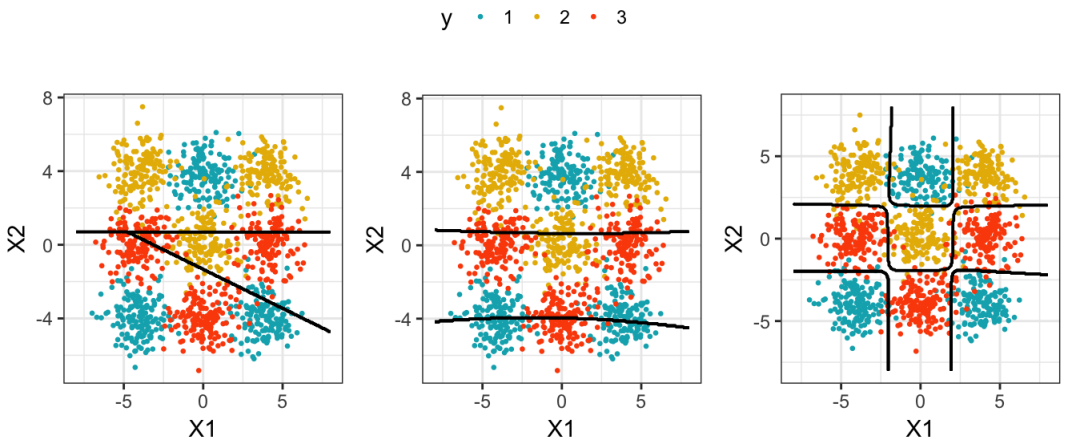

Figure 1.9: An example of discriminant cluster analysis.

Clustering is a type of data exploration . As a result, the explorer may have no or limited knowledge of the parameters of the cluster analysis that results [110]. Clustering is used in a variety of ways and the objective is to figure out everything of the following:

1. The size of the clusters.
2. The clusters' absolute and relative locations.
3. The number of clusters.
4. Cluster density that is clusters based on density are dense areas in the data space divided by sparser portions.

1.2.7 Approximate Method Augmented Lagrangian and Penalty methods

Approximation methods are a very active area in optimization. Consider the minimizing convex function $\mathcal{G}\colon \mathbb{R}^n \to \mathbb{R}$ on a convex set X. The goal of approximation methods is to replace $\mathcal{G}$ and X with approximation $\mathcal{G}^k$ and X^k. The approximation method works only if the approximation is easier than the original problem . For each iteration k we tried to find

$$x^{k+1} = \arg \min_{x \in X^k} \mathcal{G}^k(x),$$

then at the next iteration, $\mathcal{G}^{k+1}$ and X^{k+1} are generated by the approximation which depends on the new point x^{k+1} .

Many great approximation methods are based on this idea, such as polyhedral approximation, the penalty method, the augmented Lagrangian method and interior point methods. Our work focuses on the penalty and augmented Lagrangian methods [1]. In general, the existence of constraints complicates the algorithmic solution and narrows the range of viable methods in optimization problems. As a result, it's only reasonable to try to remove constraints by approximating the relevant indicator function. For example, replace restrictions with penalty functions that impose a significant cost for breaking them [15].

The linear equality constrained problem is given by

$$\begin{aligned} &\text{minimize} && \langle c, x\rangle \\ &\text{subject to} && \langle a_i, x\rangle = b_i,\ i = 1, \ldots, m, \\ & && x \in X. \end{aligned} \tag{1.2}$$

replaced above problem with a penalized version

$$\begin{aligned} &\text{minimize} && \langle c, x\rangle + \alpha^k \sum_{i=1}^{m} P_q(\langle a_i, x\rangle - b_i) \\ &\text{subject to} && x \in X. \end{aligned} \tag{1.3}$$

The scalar α^k is a positive penalty parameter, and decreasing α^k to 0, the solution x^k of the penalized problem tends to decrease the constraint violation, thereby providing an increasingly accurate approximation to the original problem. An important practical point here is that α^k should be decreased gradually, using the optimal solution of each approximating problem to start iteration of the next approximating problem .

One choice for P_q can be the quadratic penalty function, where the penalized problem (1.3)

takes the form [12, 98],

$$\begin{aligned} &\text{minimize} \quad \langle c, x\rangle + \frac{1}{2\alpha^k}\|Ax-b\|^2 \\ &\text{subject to} \quad x \in X. \end{aligned} \tag{1.4}$$

where $Ax = b$ represents the system of equation $\langle a_i, x\rangle = b_i, \;\; i = 1, \ldots, m$. The augmented Lagrangian method is a significant improvement over the penalty function approach, where we add a linear term to $P_q(y)$, involving a multiplier vector $y^k \in \mathbb{R}^n$. Then instead of problem (1.3), we solve the problem

$$\begin{aligned} &\text{minimize} \quad \langle c, x\rangle + (y^k)^T(Ax-b) + \frac{1}{2\alpha^k}\|Ax-b\|^2 \\ &\text{subject to} \quad x \in X. \end{aligned} \tag{1.5}$$

After the solution of the above problem x^k is obtained, the multiplier vector y^k is updated by some formula that seeks to approximate an optimal dual solution [6] ,such that

$$y^{k+1} = y^k + \frac{1}{\alpha^k}(Ax^k - b).$$

This is called **The first order augmented Lagrangian methods** is what it's called (the first order method of multipliers). For both inequality and equality requirements, penalty and augmented Lagrangian methods can be employed [68].

CHAPTER 2

MATHEMATICAL BACKGROUND

2.1 Introduction

In this chapter, we provide definitions, notation, and necessary results of Euclidean space. Also, we present some basic concepts and facts of real mathematical analysis.

2.2 Basic Facts and Definitions

2.2.1 Convex Sets

Definition 2.2.1. [91] Let $S \subseteq \mathbb{R}^n$. If the line segment between any two points in S lies in S, i.e.

$$\alpha x_1 + (1-\alpha)x_2 \in S, \quad \forall x_1, x_2 \in S, \forall \alpha \in [0,1]$$

then S is said to be convex set, (see Figure 2.1):

Figure 2.1: Convex and non-convex sets.

2.2.2 Convex Function

Definition 2.2.2. [12] Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set. If $f\colon S \to \mathbb{R}$ satisfies

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha) f(x_2), \quad \forall x_1, x_2 \in S, \ \forall \alpha \in [0,1],$$

then f is said to be a **convex function** on S. If the above inequality is true as a strict inequality for all $x_1 \neq x_2$ and for all $\alpha \in (0,1)$, then f is called a **strictly convex function** on S. If there is a constant $b > 0$ such that for all $x_1, x_2 \in S$, and for all

$\alpha \in [0,1]$,

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha)f(x_2) - \frac{1}{2}b\alpha(1-\alpha)\|x_1 - x_2\|^2,$$

then f is called **strongly convex function** on S. A function f is **concave** if $-f$ is convex , (see Figures 2.2 and 2.3):

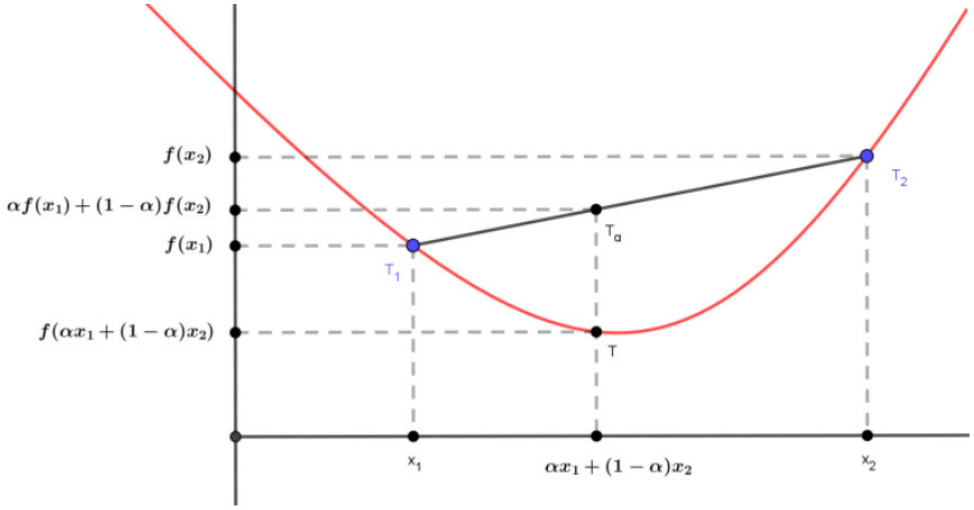

Figure 2.2: Convex function in two dimensions.

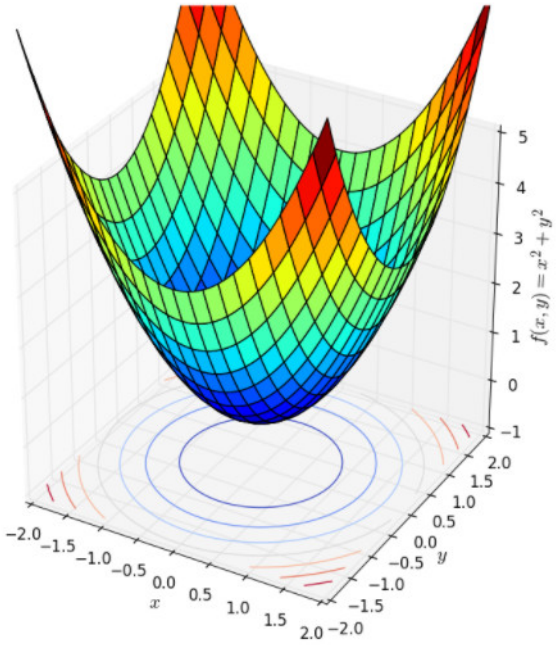

Figure 2.3: Convex function in three dimensions.

The convex analysis is important for several reasons, one of the reasons that convex analysis is a standard tool in many different fields is analogy with calculus. Another reason why convex analysis is important is that if an efficient and reliable method is required to find a good approximation of an optimal solution to an optimization problem, it is always possible to find it if that problem is convex, but it is rarely possible otherwise.

2.2.3 Epigraph and Hypograph of a Function

Let $W \subseteq \mathbb{R}^n$ be a nonempty convex set. The epigraph of a function $f : W \to \mathbb{R}$, denoted by epi (f), is a subset of $\mathbb{R}^{n+1}$ defined by:

$$\text{epi}(f) = \{(x, \delta) \mid f(x) \leq \delta, x \in W, \delta \in \mathbb{R}\}$$

The hypograph of f, denoted by $hyp(f)$, is a subset of $\mathbb{R}^{n+1}$ defined by

$$hyb(f) = \{(x, \delta) \mid f(x) \geq \delta, x \in W, \delta \in \mathbb{R}\}$$

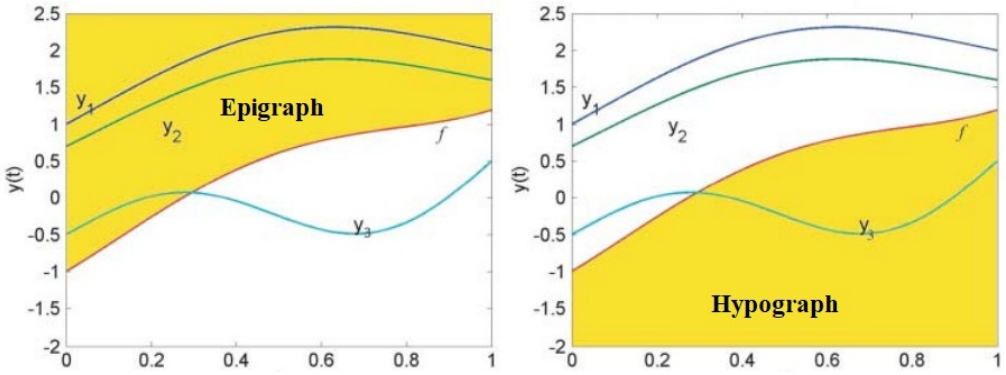

Figure 2.4: Epigraph and hypograph for function f

The Figure 2.4 show that an example of epigraph and hypograph for function f , [85].

2.2.4 Vector Norm

In this section, we introduce the Euclidean space, which means a finite dimensional vector space with an inner product $\langle \cdot, \cdot \rangle$ and the Euclidean norm defined by $\|x\| = \sqrt{\langle x, x \rangle}$ for all vectors x in the vector space.

Definition 2.2.3. [18] A function $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is an **inner product** if

(i) $\langle x, x \rangle > 0$, $\langle x, x \rangle = 0 \Leftrightarrow x = 0$ (positivity).

(ii) $\langle x, y \rangle = \langle y, x \rangle$ (symmetric).

(iii) $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$ (additivity).

(iv) $\langle \alpha x, y \rangle = \alpha \langle x, y \rangle$ for all $\alpha \in \mathbb{R}$ (homogeneity).

Definition 2.2.4. [101] A function $\| \cdot \| : \mathbb{R}^n \to \mathbb{R}$ is said to be a **norm**, if it satisfies the following properties:

(i) $\|x\| \geq 0$, $\forall x \in \mathbb{R}^n$; $\|x\| = 0$ if and only if $x = 0$.

(ii) $\|\alpha x\| = |\alpha| \|x\|$, $\forall \alpha \in \mathbb{R}$, $\forall x \in \mathbb{R}^n$.

(iii) $\|x+y\| \leq \|x\| + \|y\|$, $\forall x, y \in \mathbb{R}^n$.

The **Euclidean norm** is defined by

$$\|x\|_2 = \sqrt{x^T x} = \sqrt{\sum_{i=1}^{n} x_i^2}.$$

2.2.5 Rates of Convergence

One of the ways in which algorithms will be compared is via their rates of convergence to some limiting value. Typically, we have an iterative algorithm that is trying to find the maximum or minimum of a function and we want an estimate of how long it will take to reach that optimal value. Rates of convergence are generally defined in our context by how much information about the target function f we utilize in the algorithm's updating phase. An algorithms that require more information about f, such as derivative information, converge faster [79].

2.2.6 The Adjacency Matrix

Definition 2.2.5. [59] The adjacency matrix of a graph G with n vertices is the matrix $A \in \mathcal{M}_n$ whose entries are

$$a_{i,j} = \begin{cases} 1, \text{ if there is an edge between the } i\text{-th and } j\text{-th vertices} \\ 0, \text{ otherwise.} \end{cases}$$

The size of a graph's adjacency matrix is equal to the number of vertices, and its entries show which pairs of vertices have edges connecting them.

Example 2.2.1. [89]

Construct the adjacency matrix of the graph of the following Figure 2.5:

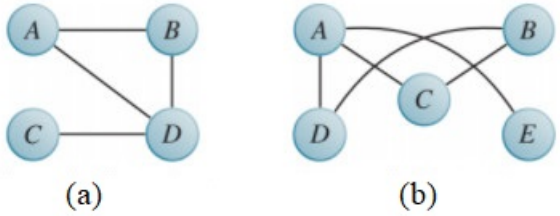

Figure 2.5: Some small graphs, the graph (a) in the left side with 4 vertices and in the right side (b) with 5 vertices.

In graph(a), since vertex A is connected to vertices B and D via edges, we place a 1 in the 2 nd and 4 th entries of the first row of the adjacency matrix. Using similar reasoning for the other rows results in the following adjacency matrix:

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

2.2.7 Symmetric Matrices

The matrices encountered most frequently in numerical optimization are symmetric.

Definition 2.2.6. [108] Let $X = [x_{ij}]$ be a square matrix. The matrix X is called **symmetric** if $X = X^T$ (i.e., $x_{ij} = x_{ji}$). Let S^n stand for the set of all of symmetric $n \times n$ matrices:

$$S^n = \left\{ X \in \mathbb{R}^{n \times n} \;\middle|\; X = X^T \right\}.$$

The notation $X \succ 0$ ($X \succeq 0$) means that X is symmetric and **positive definite** (**semidefinite**) respectively; that is for all nonzero $u \in \mathbb{R}^n$ ($u^T X u = \sum_{ij} X_{ij} u_i u_j \geq 0$ for

all $u \in \mathbb{R}^n$) [116]. Furthermore, we let S_+^n denote the set of ,:

$$S_+^n = \left\{ X \in S^n \;\middle|\; X \succeq 0 \right\}.$$

For any matrix $M \in \mathcal{S}^n$, we denote the **Frobenius norm** of M by

$$\|M\|_F = \sqrt{\langle M, M \rangle}.$$

For a number $m \in \mathbb{R}$, we label its nonnegative part by

$$m_+ = \max\{m, 0\}.$$

For vector $x \in \mathbb{R}^n$, we define the vector x_+ by $(x_+)_i = (x_i)_+$, for $i = 1, \ldots, n$; for a matrix $M \in \mathcal{S}^n$, we define

$$M_+ = U\mathrm{Diag}(\lambda_+)U^T,$$

where M has eigen decomposition $M = U\mathrm{Diag}(\lambda)U^T$, with eigenvalues $\lambda = \lambda(M) \in \mathbb{R}^n$ and orthogonal matrix $U \in \mathbb{R}^{n \times n}$.

2.2.8 Jacobian Matrices

For any basis $(r_1, \ldots, r_n)$ of M and any basis $(b_1, \ldots, b_m)$ of N, if both M and N are of finite dimension, every function $f : M \to N$ is determined by m functions $f_i : M \to \mathbb{R}$ (or $f_i : M \to \mathrm{C}$), where

$$f(x) = f_1(x)b_1 + \cdots + f_m(x)b_m$$

for every $x \in M$. we have

$$\mathrm{D}f(a)\,(r_j) = \mathrm{D}_{rj}f(a) = \partial_j f(a)$$

We have,

$$\mathrm{D}f(a)\,(r_j) = \mathrm{D}f_1(a)\,(r_j)\,b_1 + \cdots + \mathrm{D}f_i(a)\,(r_j)\,b_i + \cdots + \mathrm{D}f_m(a)\,(r_j)\,b_m$$

that is,

$$\mathrm{D}f(a)\,(r_j) = \partial_j f_1(a) b_1 + \cdots + \partial_j f_i(a) v_i + \cdots + \partial_j f_m(a) b_m$$

the linear map $\mathrm{D}f(a)$ is defined by the $m \times n$-matrix because the j-th column of the $m\times$ n-matrix representing D $f(a)$ w.r.t. the bases $(r_1, \ldots, r_n)$ and $(b_1, \ldots, b_m)$ is equal to the components of the vector $\mathrm{D}f(a)\,(r_j)$ over the basis $(b_1, \ldots, b_m)$ $J(f)(a) = (\partial_j f_i(a))$, (or $J(f)(a) = (\partial f_i/\partial x_j)\,(a)$) :

$$J(f)(a) = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1}(a) & \dfrac{\partial f_1}{\partial x_2}(a) & \ldots & \dfrac{\partial f_1}{\partial x_n}(a) \\ \dfrac{\partial f_2}{\partial x_1}(a) & \dfrac{\partial f_2}{\partial x_2}(a) & \ldots & \dfrac{\partial f_2}{\partial x_n}(a) \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial f_m}{\partial x_1}(a) & \dfrac{\partial f_m}{\partial x_2}(a) & \ldots & \dfrac{\partial f_m}{\partial x_n}(a) \end{bmatrix}$$

The Jacobian matrix of $\mathrm{D}f$ at a is the name given to this matrix. When m = n, the Jacobian of Df is the determinant, $\det(J(f)(a))$, of $J(f)(a)$. This determinant depends on $\mathrm{D}f(a)$, not on specific bases. Partial derivatives, on the other hand, provide a method for computing it. It is simple to compute the partial derivatives $\left(\dfrac{\partial f_i}{\partial x_j}\right)(a)$, for any function $f : \mathbb{R}^n \to \mathbb{R}^m$ when $M = \mathbb{R}^n$ and $N = \mathbb{R}^m$. The typical derivative is to consider the function $f_i : \mathbb{R}^n \to \mathbb{R}$ as a function of its j$-$ th parameter, keeping the others constant [14].

Definition 2.2.7. [1] Let $f : \mathbb{R}^n \to \mathbb{R}$ and $x \in \mathbb{R}^n$. Then the partial derivative of f at x

with respect to x_i is defined as

$$\frac{\partial f(x)}{\partial x_i} = \lim_{t \to 0} \frac{f\left(x + te_i\right) - f(x)}{t}$$

where e_i is i th unit vector. The gradient of f at x is defined as the column vector

$$\nabla f(x) = \begin{bmatrix} \dfrac{\partial f(x)}{\partial x_1} \\ \vdots \\ \dfrac{\partial f(x)}{\partial x_n} \end{bmatrix}$$

The Hessian matrìx is defined as the $n \times n$ symmetric matrix

$$\nabla^2 f(x) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1 \partial x_1} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}$$

The directional derivative of the function f at x in the direction d given by

$$f'(x, d) = \lim_{x \to 0^+} \frac{f(x + td) - f(x)}{t}$$

We say the function f is differentiable at x if and only if the gradient $\nabla f(x)$ exists and satisfies $\langle \nabla f(x), d \rangle = f'(x, d), \forall d \in \mathbb{R}^n$. Moreover, we say the function f is differentiable over a subset S of $\mathbb{R}^n$ if it is differentiable at every $x \in S$, and f is continuously

differentiable over S, if

$$\lim_{d\to 0} \frac{f(x+d) - f(x) - \langle \nabla f(x), d\rangle}{\|d\|} = 0, \quad \forall x \in S$$

where $\|\cdot\|$ is an arbitrary vector norm.

Definition 2.2.8. [52] **(Adjoint of a linear transformation)** Let $A_i \in \mathcal{S}^n$, for $i = 1, \ldots, m$ and define the linear transformation $\mathcal{A}\colon \mathcal{S}^n \to \mathbb{R}^m$ as $(\mathcal{A}X)_i = \langle A_i, X\rangle$, for $i = 1, \ldots, m$. Then the adjoint of $\mathcal{A}$ is defined as the unique linear transformation $\mathcal{A}^*\colon \mathbb{R}^m \to \mathcal{S}^n$ that satisfies $\langle \mathcal{A}X, y\rangle = \langle X, \mathcal{A}^* y\rangle$ for all $X \in \mathcal{S}^n$ and for all $y \in \mathbb{R}^m$. Since

$$\langle \mathcal{A}X, y\rangle = \sum_{i=1}^{m} y_i \operatorname{tr}(A_i X) = \operatorname{tr}\left(X \sum_{i=1}^{m} y_i A_i\right) = \langle X, \mathcal{A}^* y\rangle$$

we obtain $\mathcal{A}^* y = \sum_{i=1}^{n} y_i A_i$.

2.3 Basic Concepts of Optimization Programming

In this section, the basic definitions and concepts that fall within the scope of our research in optimization programming will be reviewed.

2.3.1 Definitions and Properties

In this part, we'll go over some fundamental information, and relatedness Definitions of the concept of a general optimization problem.

Definition 2.3.1. [8] **Optimization problem** is a computational problem in which the object is to find the best of all possible solutions. In other word, finding a solution in the feasible region, which has the minimum (or maximum) value of the objective function.

Definition 2.3.2. [91] **Optimal Solution** is a solution to an optimization problem which minimizes (or maximizes) the objective function. As shown in the following diagram and how the strategy to obtain the optimal solution ,(see Figure 2.6).

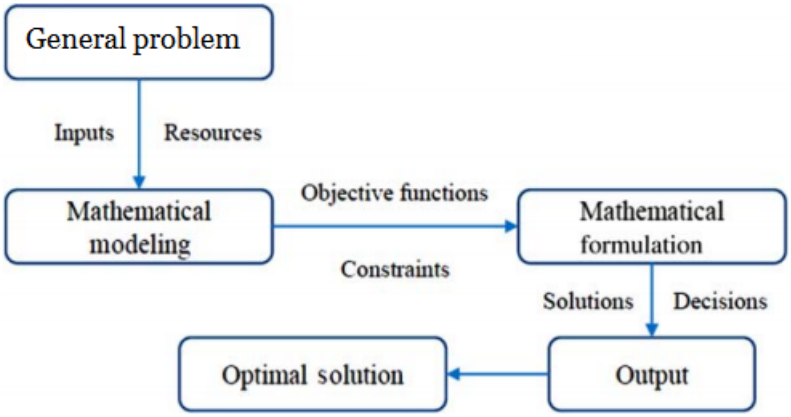

Figure 2.6: Flow chart modeling a generic problem to find the optimal solution

Definition 2.3.3. [118] An **objective function** is the output that you want to maximize or minimize. For example, the objective function is usually to maximize portfolio value while aerospace engineering the objective is often to minimize weight. As in the following figure, where the goal function of the mountain climber is to reach the highest peak, that is, the objective function is maximize, (see Figure 2.7).

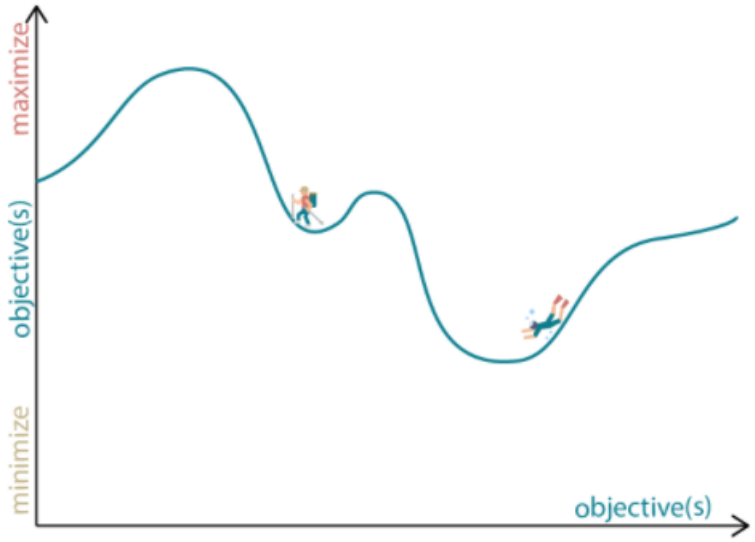

Figure 2.7: Simple example of objective function

Definition 2.3.4. [95]. **Feasible region** is the set of all feasible solutions to a problem of optimization, as shown in the figure below ,(see Figure 2.8).

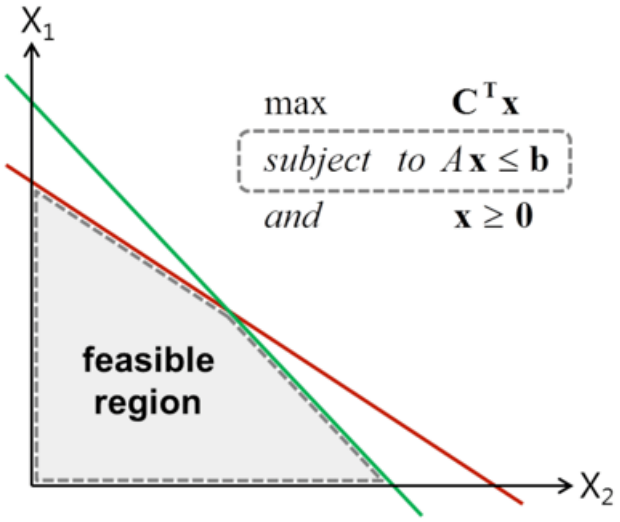

Figure 2.8: Simple example of feasible region linear programming

Definition 2.3.5. [34]. Let $S \subseteq \mathbb{R}^n$ be a nonempty set. If $f : S \to \mathbb{R}$, the argument of the minimum is the set of elements in S that achieve the global minimum in S, which is defined by :

1. $\mathbf{x}^* \in S$ is called a **global minimum point** of f over S if $f(\mathbf{x}) \geq f(\mathbf{x}^*)$ for any $\mathbf{x} \in S$,

2. $\mathbf{x}^* \in S$ is called a**strict global minimum point** of f over S if $f(\mathbf{x}) > f(\mathbf{x}^*)$ for any $\mathbf{x}^* \neq \mathbf{x} \in S$,

3. $\mathbf{x}^* \in S$ is called a **global maximum point** of f over S if $f(\mathbf{x}) \leq f(\mathbf{x}^*)$ for any $\mathbf{x} \in S$,

4. $\mathbf{x}^* \in S$ is called a **strict global maximum point** of f over S if $f(\mathbf{x}) < f(\mathbf{x}^*)$ for any $\mathbf{x}^* \neq \mathbf{x} \in S$.

Definition 2.3.6. [88]. Let $f : \mathbb{R}^n \to \mathbb{R}$ and $x^* \in \mathbb{R}^n$. Then x^* is called a **local minimimizer** of f if there is a scalar $t > 0$ such that $f(x^*) \leq f(x)$ for all $x \in \mathcal{B}(x^*, t) = \{x \in \mathbb{R}^n \mid \|x - x^*\| \leq t\}$, (see Figure 2.9).

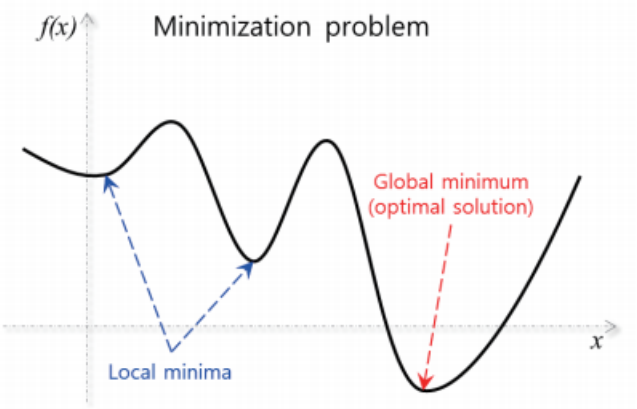

Figure 2.9: Simple example of local and global solutions in an optimization problem.

Definition 2.3.7. [116] A **polyhedral set or polyhedron** is the intersection of a finite number of closed half spaces. In other word, a subset P of $\mathbb{R}^n$ is a polyhedron if there exist a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and a vector $\boldsymbol{b} \in \mathbb{R}^m$ such that

$$P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$$

holds. The importance of polyhedra is obvious, since the set of feasible solutions of every linear programming problem is a polyhedron. Also, polyhedron is convex set. Besides, one can think of this as the feasible space (the set of all points satisfying the constraints) of a set of linear inequality constraints ,(see Figures 2.10 and 2.11):

$$\begin{aligned}
\left(\mathbf{a}^1\right)^\top \mathbf{x} &\leq b^1 \\
\left(\mathbf{a}^2\right)^\top \mathbf{x} &\leq b^2 \\
&\vdots \\
\left(\mathbf{a}^m\right)^\top \mathbf{x} &\leq b^m.
\end{aligned}$$

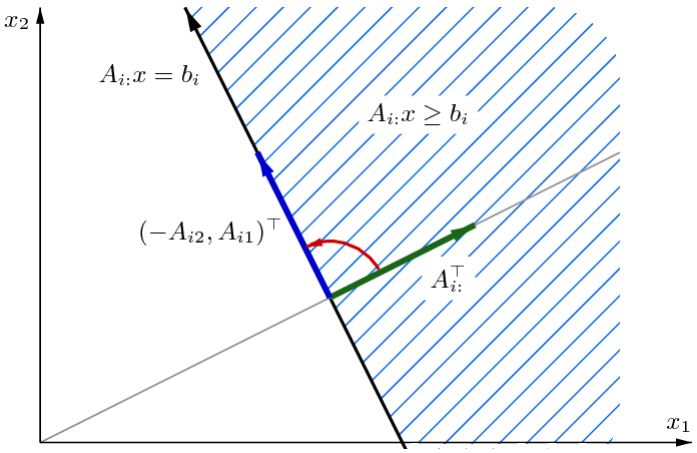

Figure 2.10: Simple example of Polyhedral

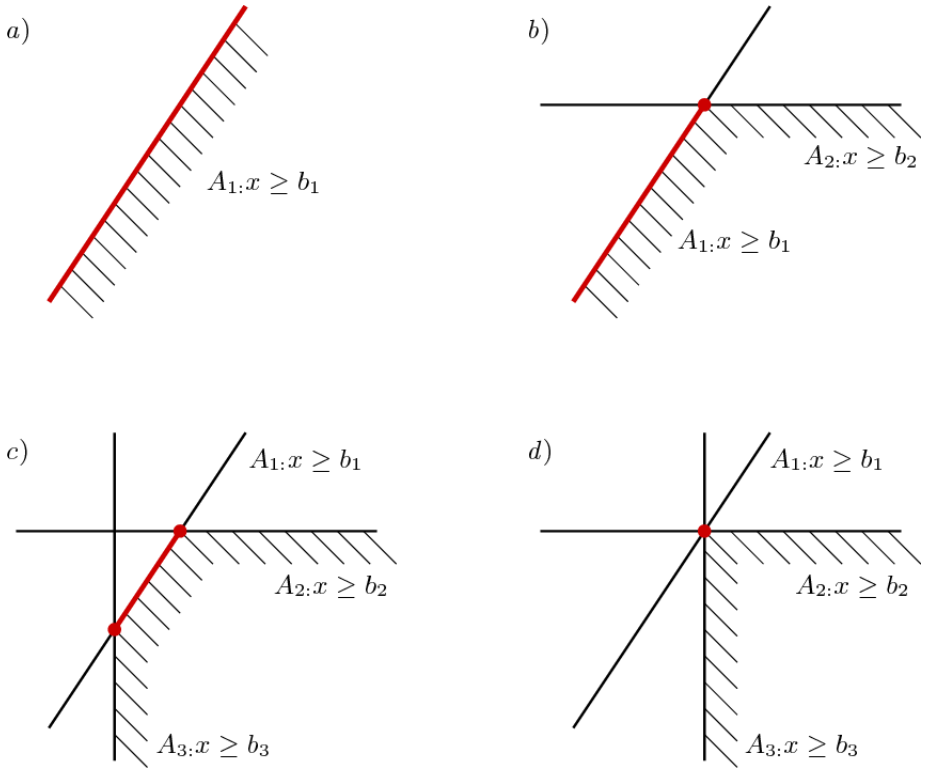

Figure 2.11: Polyhedral in many case.

2.4 Classification of the Optimization Problems

According to the type of the objective and constraint functions, the optimization problems can be classified such as continuous or discrete. In fact, in continuous

optimization the functions require to be continuous [122]. It indicates that each of the n choice variables' solution value might be any real number, while in discrete, some or all of the variables have integral values at the optimal solution. Further, the continuous optimization problem can be classified either unconstrained or constrained optimization problem [16].

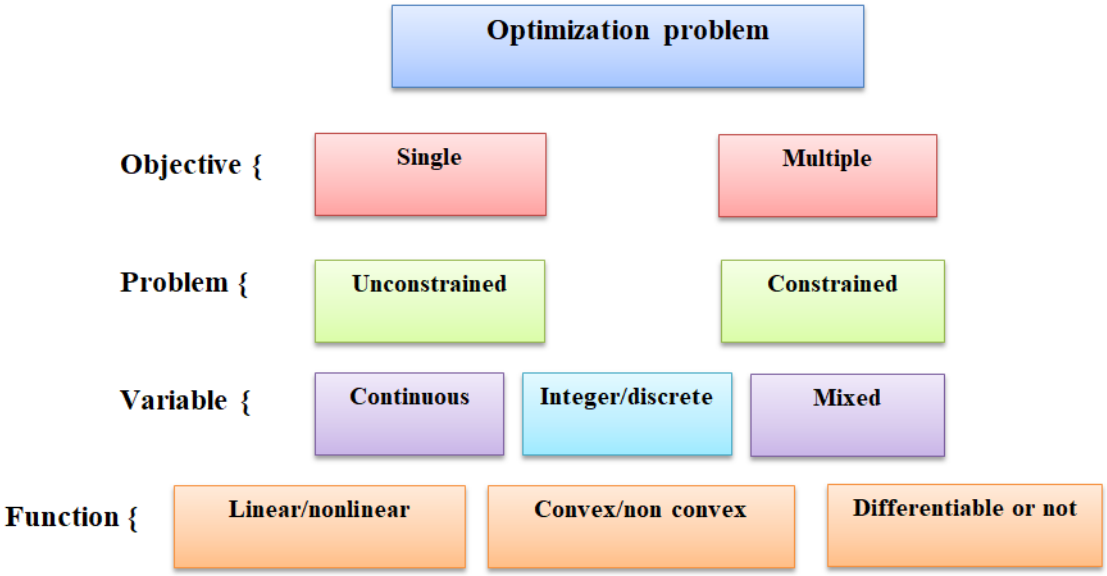

Figure 2.12: Classification of Optimization Problem.

When the set S = R^n (domain S equal to the whole space) then the problem is called an unconstrained. If S is proper subset of R^n then the problem called constrained ,(see Figure 2.12).

2.4.1 Constrained Optimization

Consider the following general constrained optimization problem:

$$\text{Max or Min} \quad f(x_1, ..., x_n)$$

Subject to

$$g_1(x_1, ..., x_n) \leq a_1, ..., g_i(x_1, ..., x_n) \leq a_i$$

$$h_1(x_1, ..., x_n) = b_1, ..., h_j(x_1, ..., x_n) = b_j$$

Where $f(x)$ is the objective function, $g(x)$ represents a constraint of inequality, and $h(x)$ represent a constraint of equality, such that f,g and h are differentiable functions and their derivatives are continuous [94].

The optimization problem of equality (or inequality) constraints, and it can be expressed as:

$$\begin{cases} \underset{x}{\text{Minimize}} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, 2, ..., M \\ & x \in X, \end{cases} \tag{2.1}$$

and

$$\begin{cases} \underset{x}{\text{Minimize}} & f(x) \\ \text{subject to} & h_j(x) = 0, \quad j = 1, 2, ..., L \\ & x \in X, \end{cases} \tag{2.2}$$

where f and h_i defined from $X \subseteq \mathbb{R}^n$ into $\mathbb{R}$ are assumed to be continuously differentiable functions. The feasible set of (2.2) is denoted by

$$C = \{x \in X | h(x) = 0\},$$

where h is the function with component functions $h_1, ..., h_L$. The Lagrangian function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is represented by:

$$\mathcal{L}(x, \gamma) = f(x) + \gamma^T h(x),$$

where $\gamma = (\gamma_1, ..., \gamma_L)^T$ is called the Lagrange multiplier vector, for optimality conditions (see [12, 13, 37]).

2.4.2 Unconstrained Optimization

Unconstrained optimization problems are openly present in many real situations. If the variables have inherent restrictions, it's usually fair to ignore them and assume that they have no bearing on the best solution. We minimize an objective function that depends on actual variables without any limitations on their values in unconstrained optimization [94]. The following is the mathematical formula of unconstrained optimization:

$$\begin{cases} \underset{X}{\text{minimize}} \quad f(x) \\ x \in \mathbb{R}^n \end{cases} \tag{2.3}$$

where x is a real vector, $n \geq 1$ elements and $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is a smooth function [94]. The first step will be to derive certain requirements that will allow to determine whether or not a point is a minimum.

2.5 Basic Concepts of Graph Theory

In this section, we review some of the basic definitions and concepts on graph theory.

Definition 2.5.1. [20] A **graph** G is defined by an ordered pair $(V(G), E(G))$, where $V(G)$ is a nonempty set whose elements are called points or vertices and $E(G)$ is a set of unordered pairs of distinct elements of $V(G)$, $\left(\text{E(G)} \subseteq \begin{pmatrix} \text{V(G)} \\ \\ 2 \end{pmatrix} \right)$. The elements of $E(G)$ are called lines or edges of the graph G. A graph G with v vertices and e edges is called a (v, e) - graph. A graph G can be represented a plane figure by drawing edge between the points (representing vertices) of the graph, (see Figure 2.13):

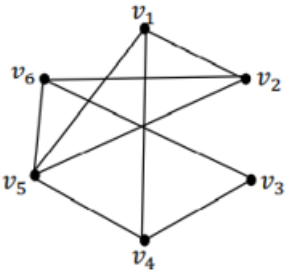

Figure 2.13: The Graph G=(v,e).

Definition 2.5.2. [48] A graph is called **connected** if any two vertices are connected by some path; otherwise it is called disconnected. It means that in a disconnected graph there always exists a pair of vertices having no path connecting them. Any disconnected graph is a union of two or more connected graphs; each such connected graph is then called a connected component of the original graph ,(see Figure 2.14):

Figure 2.14: Connected graph G_1 and disconnected graph G_2. G=(V,E)

Definition 2.5.3. [99] A graph for which each edge e has a non negative number $w(e)$ called a **weight** associated with it ,(see Figure 2.15):

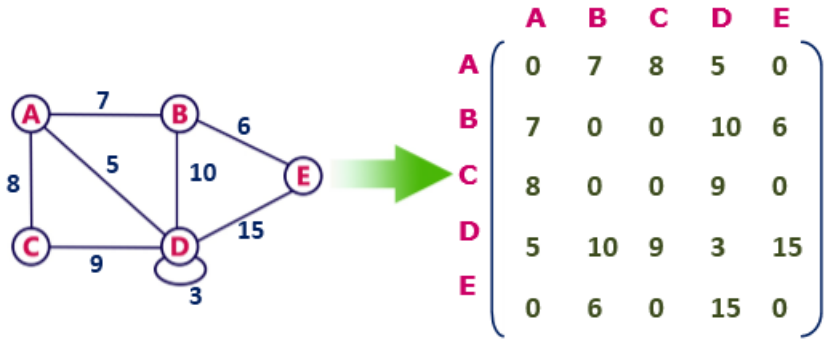

Figure 2.15: An example of a representation of weighted graphs.

Definition 2.5.4. [23] A **spanning tree** is a connected and undirected graph, with no cycles. Each tree has with n vertices. Each pair of vertices has exactly one path connecting them, creating $n-1$ edges in the tree.

The problem of finding a spanning tree of minimum (maximum) weight is called the minimum (maximum) spanning tree problem, (see Figure 2.16):

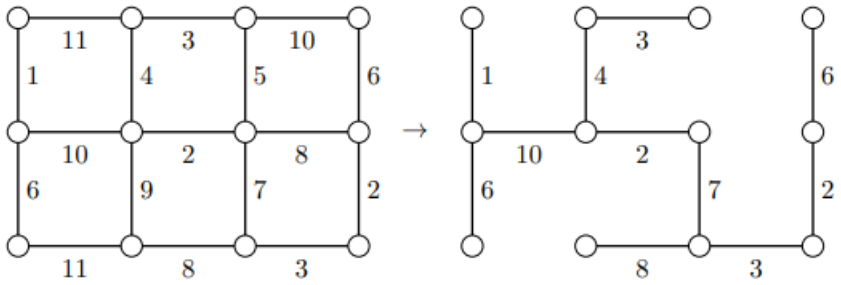

Figure 2.16: Spanning tree of weighted graph

Definition 2.5.5. [60] A graph G is **bipartite** if the vertices can be partitioned into two sets X and Y so that every edge has one endpoint in X and the other in Y. As in Figure (2.17).

Figure 2.17: Two bipartite graphs.

Definition 2.5.6. [35] A **planar graph** is graph that can be drawn without links crossing as shown in Figure (2.18).

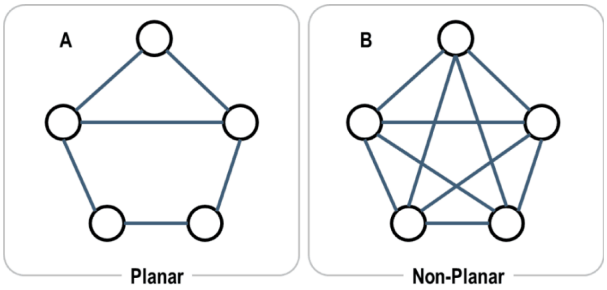

Figure 2.18: Planar and non-planar graphs

Definition 2.5.7. [10] The number of vertices in a given graph is called **order** of the graph, denoted by $|V(G)|$.

Definition 2.5.8. [57] The number of edges in a given graph is called **size** of the graph, denoted by $|E(G)|$.

Definition 2.5.9. [19] A path of length n-1, denoted by P_n, is a sequence of distinct edges $v_1v_2, v_2v_3, \ldots, v_(n-1)v_n$

Definition 2.5.10. A closed path with $v_1 = v_n$, is called a **cycle**. A cycle with n vertices is denoted by $\boldsymbol{C_n}$.

Definition 2.5.11. [23] A **regular graph** is defined as a graph that all of its vertices are of the same degree, in such case we say that the graph of degree d is a d-regular graph.

Definition 2.5.12. [48] A **complete graph** of order $n\,(K_n)$ is a regular graph of degree $n-1$ as in Figure (2.19).

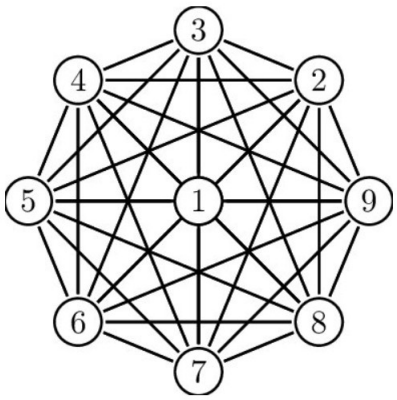

Figure 2.19: A complete graph with K_9.

2.6 Principles of Clustering Model

As shown in the figure below is the largest connected component of the popular Netscience graph [2,93], (see Figure 2.20). Two sets of nodes have been highlighted, one in blue, the other in red. Without covering any mathematical background, one may

easily guess that the red set is more likely to be considered a 'community' than the blue set. Blue nodes, which were selected at random, are scattered across the graph and share few connections with one another, whereas the red nodes appear tightly connected internally and are, for the most part, separated from the rest of the graph. Indeed, Although Figure provides an intuitive visual aid for understanding the task of graph clustering, it is oversimplified in a number of ways.

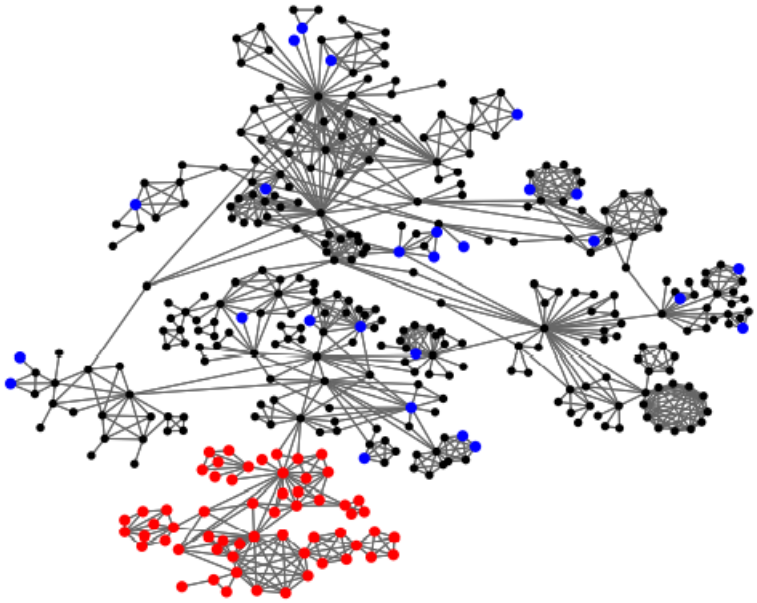

Figure 2.20: Popular Netscience Graph

Besides, real-world graphs are usually much larger and do not come with a clean two-dimensional layout that visually highlights community structure. Although is clear that the red set is a better community than the blue set, in practice it becomes very challenging to understand how to find the best partitioning of a graph into communities for a specific application.

2.7 The Linear Programming and The Semidefinite Programming

Semidefinite optimization is a branch of convex optimization that is of great theoretical and practical interest [17]. In fact, the main idea is to generalize linear programming and the associated feasible sets (polyhedra) to the case where the decision variables are symmetric matrices, and the inequalities are to be understood as matrices being positive semidefinite [116].

2.7.1 Linear Programming

The problem of minimizing a linear function with linear constraints is known as linear programming. In standard form, a linear programming problem (LP) is written as

$$(LP)\begin{cases} \text{minimize} & \langle c, x\rangle \\ \text{subject to} & \langle a_i, x\rangle = b_i, \quad i = 1, \ldots, m \\ & x \geq 0, x \in \mathbb{R}^n. \end{cases}$$

The problem data are given by the vectors $a_i \in \mathbb{R}^n$, for $i = 1, \ldots, m$, the numbers $b_i \in \mathbb{R}$, for $i = 1, \ldots, m$, and the vector $c \in \mathbb{R}^n$. Consider the primal (P) and the dual (D) standard linear programming problem: and

$$(D)\begin{cases} \text{maximize} & \langle b, y\rangle \\ \text{subject to} & A^T y \geq c \end{cases} \equiv \begin{cases} \text{maximize } \langle b, y\rangle \\ \text{subject to } \left[c - A^T y\right]_+ = 0 \end{cases}$$

One of the most surprising and valuable characteristics of linear programming is that we may correlate a matching dual problem with every LP problem. It's worth noting that we're optimizing a linear function over a polyhedron once more. The primal problem (LP-

P) and its dual problem (LP-D) have extremely natural and direct algebraic connections, as we will demonstrate (LP-D), [24].

2.7.2 The Relationships Between Primal and Dual Problem

If there is a solution to the primal problem, there is likewise a solution to the dual problem. Furthermore, the solution value for both problems is the same, i.e., the primal objective function's maximum value is exactly equal to the dual function's minimum value [124].

Example 2.7.1. Consider the following simple linear programming problem:

$$(P)\begin{cases}\text{maximize} & x_1 + 2x_2 + 4x_3 \\ \text{subject to} & x_1 + x_2 + x_3 = 9, \\ & x \geq 0.\end{cases} \tag{2.4}$$

The dual of problem (2.4) is given by

$$(D)\begin{cases}\text{minimize} & 9y_1 \\ \text{subject to} & y_1 \geq 1, \\ & y_1 \geq 2, \\ & y_1 \geq 4.\end{cases} \tag{2.5}$$

The optimal solution of problem (2.4) is $x_1 = x_2 = 0$, $x_3 = 9$ and the optimal value is $x_1 + 2x_2 + 4x_3 = 36$; an optimal solution of problem (2.5) is $y_1 = 4$, and the optimal value is $9y_1 = 36$.

The following table (2.1) briefly shows the relationship between primal and dual Problem:

Primal	Dual
$\leq$ constraint	variable ≥ 0
$\geq$ constraint	variable ≤ 0
$=$ constraint	variable ≥ 0 or ≤ 0
max	min
right-hand side	objective coefficient
variable ≥ 0	$\geq$ constraint
variable ≤ 0	$\leq$ constraint
variable ≥ 0 or ≤ 0	$=$ constraint
objective coefficient	right-hand side
number of constraints	number of variables
number of variables	number of constraints

Table 2.1: The Relationships between Primal and Dual Problem.

2.7.3 Semidefinite Programming

The semidefinite program in equational form is the following kind of optimization problem:

$$(SDP)\begin{cases}\text{minimize} & \langle C, X\rangle \\ \text{subject to} & \langle A_i, X\rangle = b_i, \quad i = 1, \ldots, m, \\ & X \succeq 0, \quad X \in \mathcal{S}^n.\end{cases} \tag{2.6}$$

where $\mathcal{S}^n$ is the set of $n \times n$ symmetric matrices, and the problem data is given by the matrices $A_i \in \mathcal{S}^n$, for $i = 1, \ldots, m$, the numbers $b_i \in \mathbb{R}$, for $i = 1, \ldots, m$, and the matrix $C \in \mathcal{S}^n$. The inner product of C and X in $\mathcal{S}^n$ is defined as $\langle C, X\rangle = \sum_{ij} C_{ij}X_{ij} = \text{trace}(CX)$. The notation $X \succ 0 (X \succeq 0)$ means that X is symmetric and positive definite (semidefinite); that is, $X = X^T$ and $u^T X u = \sum_{ij} X_{ij}u_i u_j > 0$ for all

nonzero $u \in \mathbb{R}^n$ $\left(u^T X u = \sum_{ij} X_{ij} u_i u_j \geq 0 \text{ for all } u \in \mathbb{R}^n \right)$. Furthermore, we let $\mathcal{S}_+^n$ be the set of positive semidefinite $n \times n$ symmetric matrices.

The dual problem of the SDP problem (2.6) is given by

$$(SDD) \begin{cases} \text{maximize} & \langle b, y \rangle \\ \text{subject to} & \sum_{i=1}^{m} y_i A_i + S = C, \\ & S \succeq 0. \end{cases} \tag{2.7}$$

We define the optimal values of the primal problem (2.6) as

$$(P) : p^* := \inf \{ \langle C, X \rangle : \langle A_i, X \rangle = b_i, i = 1, \ldots, m, X \succeq 0 \}$$

which has an associated Lagrangian dual problem [1]. The augmented Lagrangian function is given by

$$\mathcal{L}_{\alpha p}(X, \gamma) = \langle C, X \rangle + \gamma^T (b - \mathcal{A}X) + \frac{1}{2\alpha} \|b - \mathcal{A}X\|_F^2.$$

The Lagrangian function for (2.6) is given by

$$\begin{aligned} \mathcal{L}(X, \gamma) &= \langle C, X \rangle + \langle \gamma, b - \mathcal{A}X \rangle \\ &= b^T \gamma + \langle C - \mathcal{A}^* \gamma, X \rangle . \end{aligned} \tag{2.8}$$

Also, we define the optimal values of the dual problem (2.7) as

$$(D) : d^* := \sup_{y,S} \left\{ \langle b, y \rangle \ : \sum_{i=1}^{m} y_i A_i + S = C, S \succeq 0, y \in \mathbf{R}^m \right\}.$$

The augmented function for the dual to the linear constraints is defined as

$$\mathcal{L}_{\alpha d}(X, \gamma, S) = \langle b, \gamma \rangle + \langle X, \mathcal{A}^* \gamma + S - C \rangle + \frac{1}{2\alpha} \|\mathcal{A}^* \gamma + S - C\|_F^2,$$

where the adjoint operator $\mathcal{A}^* y = \sum_{i=1}^{m} y_i A_i$, $\alpha > 0$ satisfies $\langle A^* y, X \rangle = \langle y, \mathcal{A}X \rangle$, $X \in S^n$ and $\gamma \in \mathbb{R}^n$ is the vector of Lagrangian multipliers and $\alpha > 0$ is the penalty parameter. The duality theory of SDP is weaker than that of LP. One still has the familiar weak duality property: feasible X, y, S satisfy

$$\mathrm{Tr}(CX) - b^T y = \mathrm{Tr}\left(\left(S + \sum_{i=1}^{m} y_i A_i\right) X\right) - \sum_{i=1}^{m} y_i \,\mathrm{Tr}\,(A_i X) = \mathrm{Tr}(SX) \geq 0$$

where the inequality follows from $X \succeq 0$ and $S \succeq 0$. In other words, the duality gap is nonnegative for feasible solutions. The solutions (X, y, S) with zero duality gap

$$\mathrm{Tr}(CX) - b^T y = \mathrm{Tr}(SX) = 0$$

are optimal. For LP, if either the primal or the dual problem has an optimal solution, then both have optimal solutions [40], and the duality gap at optimality is zero. This is the strong duality property. The SDP case is more subtle: One problem may be solvable and its dual infeasible, or the duality gap may be positive at optimality, etc. The existence of primal and dual optimal solutions is guaranteed if both (P) and (D) allow positive definite solutions, [128].

Example 2.7.2. [1]

The semidefinite programming form $n = 3$ and $m = 2$. Define the following matrices:

$$A_1 = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 4 & 3 \\ 0 & 3 & 6 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 & 4 & 0 \\ 4 & 7 & 1 \\ 0 & 1 & 5 \end{bmatrix}, \quad C = \begin{bmatrix} 2 & 0 & 2 \\ 0 & 6 & 1 \\ 2 & 1 & 8 \end{bmatrix}, \text{ and } b = \begin{bmatrix} 4 \\ \\ 7 \end{bmatrix}$$

then X is the 3×3 symmetric matrix

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}$$

$$C \cdot X = 2x_{11} + 0x_{12} + 4x_{13} + 6x_{22} + 2x_{23} + 8x_{33}$$

since X is symmetric. Then the SDP can be written as the standard SDP problem is given by

$$(eSDP) \begin{cases} \text{Minimize} & 2x_{11} + 0x_{12} + 4x_{13} + 6x_{22} + 2x_{23} + 8x_{33} \\ \text{subject to} & 2x_{11} + 2x_{12} + 0x_{13} + 4x_{22} + 6x_{23} + 6x_{33} = 4, \\ & x_{11} + 8x_{12} + 0x_{13} + 7x_{22} + 2x_{23} + 5x_{33} = 7 \\ & X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \succeq 0, \quad X \in \mathcal{S}^n \end{cases}$$

2.7.4 The Importance of Semidefinite Programming

There are many reasons for **semidefinite programming problems** to be significant, including [31]:

1. SDP covers significant groups of problems, such as linear and quadratic programming (LP and QP).

2. In combinatorial optimization there are many applications for SDP such as approximation theory, system and control theory, and mechanical and electrical engineering .

3. Interior point algorithms for SDP have been researched intensely in the 1990s,

explaining the resurgence in research interest in SDP.

2.7.5 Positive Semidefinite Matrices

A symmetric matrix $A \in \mathrm{S}^n$ is said to be positive semidefinite (PSD) if the associated quadratic form is non-negative, [58] i.e.,

$$x^\top Ax \geq 0, \forall x \in \mathrm{R}^n$$

and A is said to be positive definite if

$$x^\top Ax > 0, \forall x \neq 0, x \in \mathrm{R}^n$$

To denote a symmetric positive semidefinite (resp. positive definite) matrix, we use the notation $A \succeq 0$ (resp. $A > 0$). We say that A is negative semidefinite, written $A \preceq 0$, if $-A \succeq 0$, and likewise A is negative definite, written $A < 0$, if $-A > 0$. It can immediately be seen that a positive semidefinite matrix is actually positive definite if and only if it is invertible. The set of real positive semidefinite matrices in $\mathrm{R}^{n,n}$ is denoted by

$$S_+^n = \{A \in S^n : A \succeq 0\}$$

Similarly, S_{++}^n denotes the set of positive definite matrices in R^n.

Example 2.7.3. [132]

A simple example for positive semidefinite matrix can be represented by the following A matrix :

$$A = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix},$$

Clear that the condition for the matrix to be symmetric is already satisfied. Now we need to calculate the determinant of the following matrices

$$\begin{pmatrix} 2 \end{pmatrix}, \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \text{ and } \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}$$

$$\begin{vmatrix} 2 \end{vmatrix} > 0, \begin{vmatrix} 2 & -1 \\ -1 & 2 \end{vmatrix} = 3 > 0, and \begin{vmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{vmatrix} = 0$$

,

since all the determinant which calculated are grator than or equal to zero, hence we can say the matrix A is positive semidefinite.

2.7.6 Relaxation for Semidefinite Programming

Let's start with a review of the notion of convex relaxations. They are one of the most powerful approaches for develop polynomial time approximation algorithms for NP-hard optimization problems like Chromatic Number, MAX-CUT, and Minimum Vertex Cover, among others [26]. For these problems, approximation methods are created by rewriting the problem as an integer program. The integer program is then relaxed to a convex program, such as a linear program (LP) or a semidefinite program, which can be solved in polynomial time(SDP). The solution to the combinatorial problem is then found by devising a (potentially randomized) polynomial-time method to transform the

solution of such a convex relaxation into an integer solution for the combinatorial problem, a process known as "rounding", [69].

However, semidefnite programming is not a recent concept of combinatorial optimization. In fact, semidefinite programming has recently gained popularity as a method for developing more efficient algorithms for approximating hard combinatorial optimization problems and, more broadly, polynomial optimization problems, which include optimizing a polynomial objective function over a simple closed semi-algebraic set. Since its introduction in the 1990s, the semidefinite relaxation approach has sparked a lot of interest in combinatorial optimization. Finally, semidefinite relaxation is now recognized as an effective method and have close bounds for a wide range of complicated problems and the strategy would be to replace a binary vector variable with a continuous matrix variable, [1].

2.8 NP hardness

A polynomial-time algorithm is an algorithm with worst-case running time $O\left(n^d\right)$, where n denotes the input size and d is a constant (independent of n). An oversimplification of the "easy vs. hard" dichotomy proposed by the theory of NP-hardness is: easy $\leftrightarrow$ can be solved with a polynomial-time algorithm; hard $\leftrightarrow$ requires exponential time in the worst case [71]. For example, consider the following algorithm in Figure (2.21) specifically at the second step, where there is no direct idea to choose the solution, since the polynomial contains multiple variables, so it is non-deterministic algorithm for searching. Which leads to wasting time in finding the optimal solution, and therefore the condition of finding the optimal solution that aims to reach speed and accuracy in the solution has not been fulfilled .

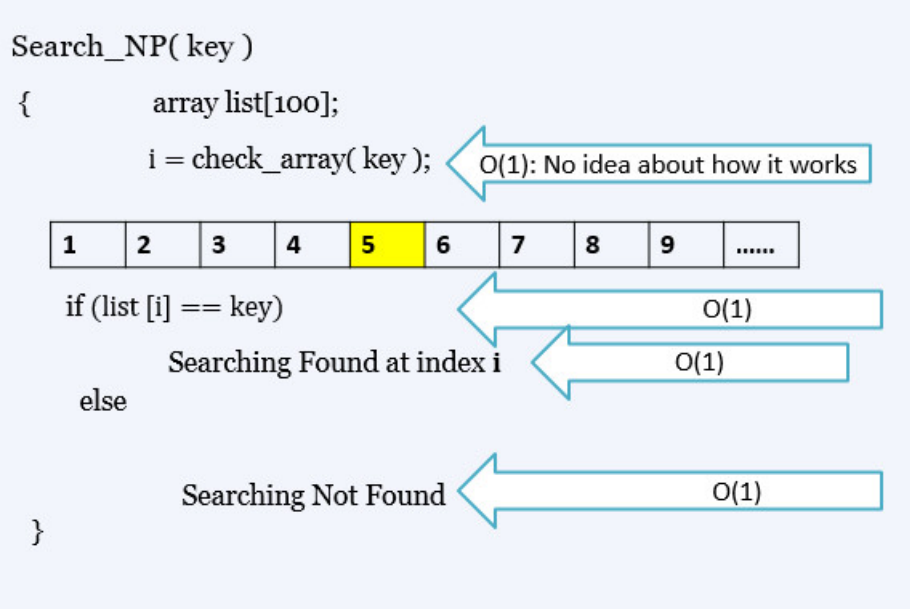

Figure 2.21: Non-deterministic algorithm for searching

The theory of NP-hardness defines "easy" problems as those solvable by a polynomial-time algorithm, or equivalently by an algorithm for which the solvable input size (for a fixed time budget) scales multiplicatively with increasing computational power. A computational problem is polynomial-time solvable if there is a polynomial-time algorithm that solves it correctly for every input. The identification of "easy" with "polynomial-time solvable" is imperfect; a problem might be solved in theory (by an algorithm that technically runs in polynomial time) but not in reality (by an empirically fast algorithm), or vice versa [127]. Polynomial-time solvability has been unreasonably effective at classifying problems as "easy" or "hard" in a way that accords with empirical experience. With a half-century of evidence behind us, we can confidently say that natural polynomial-time solvable problems typically can be solved with practical general-purpose algorithms, and that problems believed to not be polynomial-time solvable typically require significantly more work and domain expertise [109]. Finally, there are many examples of NP-hardness problem in our world such as Max-Cut problem, K-CLUSTER PROBLEM , Max-Independent-Set (MIS) problem and also has several applications in our lives today, such as the traveling

salesman problem (TSP) and other problems related to the complex network , [4].

2.8.1 The Structure of Hardness Problem

A polynomial-time algorithm would solve thousands of problems that have resisted the efforts of tens (if not hundreds) of thousands of brilliant minds over many decades. In fact, the NP hardness theory shows that thousands of arithmetic problems are convincing variations of the same problem, all of which have identical arithmetic destinies. The trying to devise a polynomial-time algorithm for an NP-hard problem is the attempting to also come up with such algorithms for these thousands of related problems. There are various types of difficult and important problems that can be classified according to the method of solution and algorithms used , [56] . A decision problem can be assigned as NP if it can be solve in polynomial time on a nondeterministic machine (or with a nondeterministic algorithm). The abbreviation NP stands for non-deterministic polynomial time (difficult to solve, easy to check a given answer). The non-deterministic Turing machine uses an algorithm which consists of two phases. The first phase is based on a guess of the solution which can be generated in a non-deterministic way. The second phase verifies the guess to be the solution to the problem by using a polynomial-time deterministic algorithm. When there is at least one polynomial-time method to solve a decision problems, it is deemed to be in the P-class, [130]. In this case, the algorithm's solution time is constrained by a polynomial in n, where n is the length of the input. A problem is NP-hard, but not necessarily a decision problem, if any other NP problem can be reduced to the NP-hard problem in polynomial time by a deterministic Turing computer. The NP-complete (NPC) problems are the most difficult in NP because, unless P=NP, a polynomial-time method to solve them is unlikely to exist. However, asymptotically faster algorithms for NP problems can exist if P=NP. Finally, the K-CLUSTER PROBLEM is NP-hard, so there are no polynomial-time K-CLUSTER algorithms for generic graphs [82]. The connections between various problem classes are depicted in the diagram below,(see Figure 2.22).

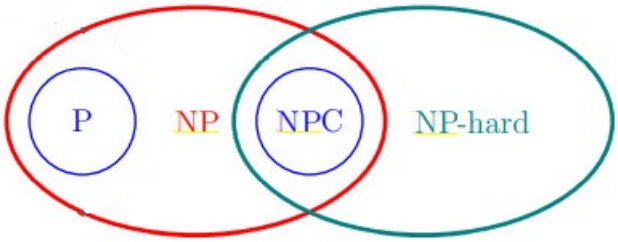

Figure 2.22: The relationship between Classification of NP-Hardness

2.9 The Quadratic Programming

2.9.1 Introduction

One of the most remarkable developments of the present century is the nonlinear optimization, in which perhaps the most important is quadratic optimization. History of quadratic optimization is very old but its formal form of study is recent for NP hard problems. It constitutes a special class of non-linear optimization problem [30]. This type of problem is NP-hard and to find its exact optimal solutions, we require the use of considerable computational resources. The development of efficient hybrid techniques, combining in a suitable way the best features of different approaches (exact or approximate) is the actual direction, in which many researchers devote their efforts to solve successfully various hard practical problems [74]. Several outstanding textbooks have been published addressing different facts of non-linear optimization and its particular case "quadratic optimization".

2.9.2 The Form of Quadratic Programming Problem

The optimization problem assumes the form

$$\begin{cases} \text{minimize } f(x) = \alpha_0 + \gamma^T x + x^T Q x \\ \\ \text{subject to: } \alpha^T x \geq \beta, x \in R^n \end{cases}$$

where

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_{11} & \alpha_{22} & \dots & \alpha_{1q} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2q} \\ \vdots & \vdots & & \vdots \\ \alpha_{n1} & \alpha_{n2} & \dots & \alpha_{nq} \end{bmatrix}$$

$$\boldsymbol{\beta}^T = \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_q \end{bmatrix}$$

$$\boldsymbol{\gamma}^T = \begin{bmatrix} \gamma_1 & \gamma_2 & \cdots & \gamma_n \end{bmatrix}$$

and **Q** is a positive definite or semidefinite symmetric square matrix, then the constraints are linear and the objective function is quadratic. Such an optimization problem is said to be a quadratic programming (QP) problem . A typical example of this type of problem is as follows ,where $f : R^n \to R$:

$$\begin{cases} \text{Maximize } f(\mathbf{x}) = \frac{1}{2}x_1^2 + \frac{1}{2}x_2^2 - x_1 - 2x_2 \\ \text{subject to} \quad c_1(\mathbf{x}) = 6 - 2x_1 - 3x_2 \geq 0 \\ \qquad\qquad\quad c_2(\mathbf{x}) = 5 - x_1 - 4x_2 \geq 0 \\ \qquad\qquad\quad c_3(\mathbf{x}) = x_1 \geq 0 \\ \qquad\qquad\quad c_4(\mathbf{x}) = x_2 \geq 0, \end{cases} \tag{2.9}$$

The quadratic program (QP) is concisely stated as follows:
Given constants A $\in \mathrm{R}^{\mathrm{n}\times\mathrm{n}}$, b $\in \mathrm{R}^{\mathrm{n}}$, c $\in$ R, and variable x $\in \mathrm{R}^{\mathrm{n}}$, minimize $f(x) = \frac{1}{2}x^T Ax + b^T x + c$ subject to the linear inequality constraints $x \geq 0$ and the number of linear inequality constraints is $n + m$. That is the linear inequalities define an intersection of half spaces. That is the intersection can be empty, in which case the QP does not have a solution. For a nonempty intersection that is unbounded and with no additional constraints on A, it is possible the Qp has no solution. If the nonempty intersection is a bounded set, that set is necessarily convex. The polynomial f is continuous and defined on a closed bounded set, which guarantees that f attains both a minimum and a maximum on the set. If $x \in R^n$ is a local maximum of the QP, then there exists $y \in R^m$ such that (x, y) satisfies the KareshKuhn-Tucker (KKT) conditions. The KKT conditions are necessary for the existence of a local maximum. When A is positive semidefinite, the KKT conditions are also sufficient for the existence of a local maximum [76].

2.10 Algorithm Design

An algorithm is a series of instructions for performing calculations in order to solve a problem and is programmed to produce results for any relevant input given a set of specific instructions. An algorithm is a finite collection of clear instructions that, given some set of beginning conditions, may be performed in a specified sequence to achieve a specific objective and that has a recognized set of end conditions. An algorithm is an attempt to devise a mathematical formula for solving a real-world problem in the most efficient

manner feasible. This recipe used to create a more reusable and generic mathematical approach that can be used to solve a broader range of related problems, [63],(see Figure 2.23).

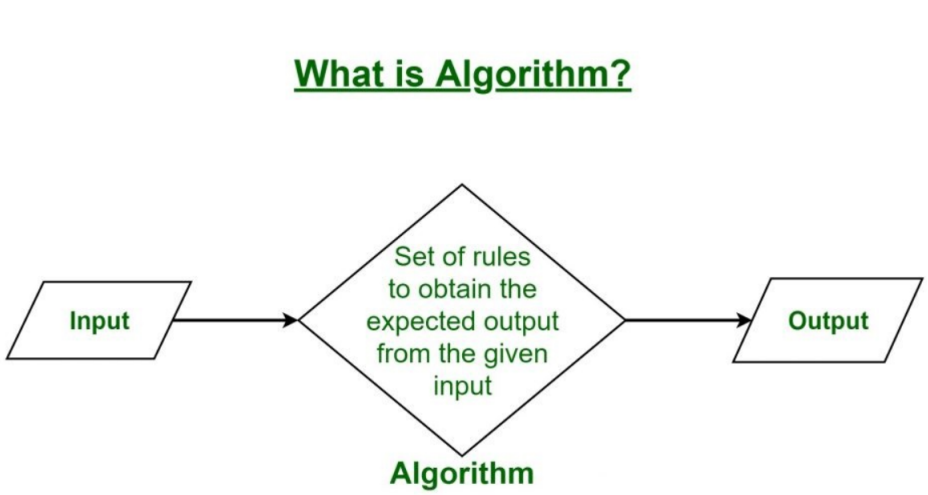

Figure 2.23: Flow chart of algorithm structure.

2.10.1 Advantages of Optimization Algorithm

According to the Figure (2.24), it depicts the standard engineering design optimization process. The designer's job is to create a problem specification that defines the conditions, constants, goals, and constraints that must be met. In some cases, the designer also provides the optimization algorithm with a basic configuration or initial design stage [63].

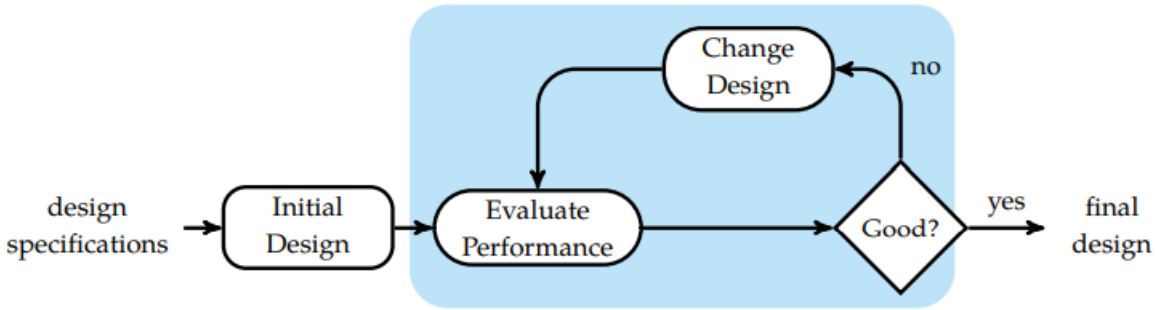

Figure 2.24: The method of design optimization. We want to simplify the blue-highlighted optimization process.

An optimization approach to mathematical design has many benefits. First and foremost, the optimization approach includes a rational and structured architecture method. Optimization algorithms, when used correctly, will help reduce the risk of human error in design. When it comes to mathematical design, intuition can be deceiving; it's always easier to optimize based on evidence. Optimization can help speed up the design process, particularly where a protocol can be written once and only used for different problems. Humans also imagine and reason for traditional engineering approaches in two or three dimensions. Modern optimization methods, can be used to solve problems involving millions of variables and constraints [92].

2.10.2 Algorithmic Strategies for NP-Hard Problems

Suppose there is a computational problem on which the success of an important project depends. The ambition was to get to the solution or get as close to it as possible. Despite many attempts and great efforts to design algorithms and review all accessible methods, it was shocking that all these efforts failed to find a solution, [47]. The problem lies not in the ingenuity of the attempts, but in the fact that the problem is NP-hard. Therefore, all efforts and time should not be given up and not be considered a failed attempt, but how to use those steps to find ways to find a solution. From this point on, the work was to find a strategy for designing algorithms that would lead to solving NP-hard problem. NP-hard problems are ubiquitous and can often (but not always) be solved in practice, at least approximately, through sufficient investment of resources and algorithmic sophistication. NP-hardness throws down the gauntlet to the algorithm designer and the expectation for the most feasible input but it is not a general-purpose and always-fast algorithm for an NP-hard problem, like shortest paths, or sequence alignment. Therefore, within an algorithmic strategy, three characteristics must be present within the formulation technique (assuming the P $\neq$ NP), [71]:

1. **General-purpose :** the algorithm accommodates all possible inputs of the computational problem.

2. **Correct approach :** for every input, the algorithm correctly solves the problem.

3. **Faster input :** for every input, the algorithm runs in polynomial time.

2.10.3 Compromising on Generality

One strategy for making progress on an NP-hard problem is to give up on general-purpose algorithms and focus instead on special cases of the problem relevant. In the best-case scenario, the identify domain-specific constraints on inputs and design an algorithm that is always correct and always fast on this subset of inputs. In problem of weighted independent set, the input is an undirected graph $G = (V, E)$ and a nonnegative weight w_v for each vertex $v \in V$; the goal is to compute an independent set $S \subseteq V$ with the maximum-possible sum $\sum_{v \in S} w_v$ of vertex weights, where an independent set is a subset $S \subseteq V$ of mutually non-adjacent vertices (with $(v, w) \notin E$ for every $v, w \in S$). For example, if edges represent conflicts (between people, courses, etc.), independent sets correspond to conflict-free subsets, [7].

The second algorithmic strategy, which is particularly popular in time-critical applications, is to insist on generality and speed at the expense of correctness. Algorithms that are not always correct are sometimes called heuristic algorithms. Ideally, a heuristic algorithm is "mostly correct." This could mean one or both of two properties "Relaxations of Correctness":

1. The algorithm is correct on "most" inputs.

2. The algorithm is "almost correct" on every input.

The second property is easiest to interpret for optimization problems, in which the goal is to compute a feasible solution (like a traveling salesman tour) with the best objective function value (like the minimum total cost). "Almost correct" then means that the algorithm outputs a feasible solution with objective function value close to the best possible, [109].

2.10.4 Compromising on Worst-Case Running Time

The final strategy is appropriate for applications in which you cannot afford to compromise on correctness and are therefore unwilling to consider heuristic algorithms. Every correct algorithm for an NP-hard problem must run in super-polynomial time on some inputs (assuming the P $\neq$ NP). The goal is to design an algorithm that is as fast as possible -at a minimum, one that improves dramatically on naive exhaustive search. This could mean one or both of two things "Relaxations of Polynomial Running Time", [106]:

1. The algorithm typically runs quickly (for example, in polynomial time) on the inputs that are relevant to your application.

2. The algorithm is faster than exhaustive search on every input.

2.10.5 Summary

From the above, there are three facts about NP-hard problems :

1. Ubiquity: Practically relevant NP-hard problems are everywhere.

2. Intractability: Under a widely believed mathematical conjecture, no NP-hard problem can be solved by any algorithm that is always correct and always runs in polynomial time.

3. Not a death sentence: NP-hard problems can often (but not always) be solved in practice, at least approximately, through sufficient investment of resources and algorithmic sophistication.

2.11 Duality

Converting a problem into a dual one is an easier approach to finding a solution is an extremely helpful tool. For instance, when solving differential equations we often use

Fourier or Laplace transforms to convert one system of differential equations into another one that's simpler to solve. The duality theory is one of the most fascinating topics in linear programming. Every LPP is linked to another LPP called dual that uses the same data and produces similar optimum solutions. The fact that these two problems are duals of one other is a boon. One is referred to as primordial, while the other is referred to as dual. The significance of the duality idea may be attributed to two factors. To begin, if the primal has a big number of constraints and a small number of variables, changing it to a dual problem and solving it can greatly reduce the computing time. Second, interpreting the dual variables from a cost or economic standpoint is highly important in making future judgments about the activities that are being planned [40]. The dual problem's answer gives a lower bound on the primal (minimization) problem's solution. However, the optimal values of the primal and dual problem's do not have to be equivalent in most cases. The duality gap is the name given to their disparity. Under a constraint qualifying condition, the duality gap for convex optimization problems is zero. The primal and dual problems are linked by two important findings. The first is known as "weak"duality, and it asserts that primal objective values set boundaries for dual objective values, and vice versa. This feature of weak duality may be used to nonlinear optimization problems and other more broad situations. While, the second, known as "strong" duality, says that the primal and dual problems' optimum values are equivalent, if they exist. There may not be a strong duality result for nonlinear situations [3]. Details will be clearer in the following two theorems:

Theorem 2.11.1. ***Duality Theorem*** *[16] Let x be a feasible point for the primal problem in standard form, and let y be a feasible point for the dual problem. Then*

$$z = c^T x \geq b^T y = w$$

Proof. The constraints for the dual show that $c^T \geq y^T A$. Since $x \geq 0$,

$$z = c^T x \geq y^T A x = y^T b = b^T y = w.$$

□

Theorem 2.11.2. *[34]* ***Strong Duality Theorem*** *Consider a pair of primal and dual linear programs. If one of the problems has an optimal solution then so does the other, and the optimal objective values are equal.*

2.11.1 The Lagrangian Duality

Duality is essential in optimization. The term dual problem usually refers to the dual Lagrangian problem. Lagrangian duality theory is about solving an optimization problem by finding a bound. Lagrangian duality gives lower or upper bounds to the original problem and can be used to evaluate how far we are from optimality [22]. In our research, we applied Lagrangian duality for SDP to get the high-quality bounds. The standard form of the nonlinear optimization problem is given by

$$(P)\begin{cases} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, I, \\ & g_i(x) = 0, \quad i = 1, \ldots, E, \\ & x \in X. \end{cases} \tag{2.10}$$

we define the Lagrangian function as

$$\mathcal{L}(x, \beta, \gamma) = f_0(x) + \sum_{i=1}^{I} \beta_i f_i(x) + \sum_{i=1}^{E} \gamma_i g_i(x),$$

where $\beta \in \mathbb{R}_+^I$ and $\gamma \in \mathbb{R}^E$ are the dual variables (Lagrange multipliers). Then we define the Lagrangian dual function to be

$$\psi(\beta, \gamma) = \min_{x \in X} \ \mathcal{L}(x, \beta, \gamma).$$

We denote the optimal value of the Lagranian dual problem by d^*, the optimal value of primal problem by p^*, and x^* is an optimal solution of (P). The Lagrange dual problem with dual variables (β, γ) is given by

$$(D) \begin{cases} \text{maximize} & \psi(\beta, \gamma) \\ \text{subject to} & \beta \geq 0. \end{cases} \tag{2.11}$$

The dual function provide lower bounds on the optimal value p^* of the problem (2.10); that is, for any $\beta \geq 0$ and any γ we have

$$\psi(\beta, \gamma) \leq p^*.$$

To prove this property, let $\beta \geq 0$ and γ be given, and let x be a feasible point for the problem (2.10) (i.e., $f_i(x) \leq 0$, $\forall i$, $g_i(x) = 0$, $\forall i$, and $x \in X$). Then we have

$$\sum_{i=1}^{I} \beta_i f_i(x) + \sum_{i=1}^{E} \gamma_i g_i(x) \leq 0$$

since it is clear each term in the first sum is nonpositive, and each term in the second sum is zero. Thus,

$$\mathcal{L}(x, \beta, \gamma) = f_0(x) + \sum_{i=1}^{I} \beta_i f_i(x) + \sum_{i=1}^{E} \gamma_i g_i(x) \leq f_0(x).$$

Therefore,

$$\psi(\beta, \gamma) = \min_{x \in \Omega} \ \mathcal{L}(x, \beta, \gamma) \leq \mathcal{L}(x^*, \beta, \gamma) \leq f_0(x^*) \leq p^*,$$

for every feasible point x^*, which proves the above property. Since we have a lower bound that depends on the parameters β, γ an important question is: What is the best lower bound that can be obtained from the Lagrangian dual function? This leads to the Lagrangian dual optimization problem (2.11).

Definition 2.11.1. [22] **(Karush-Kuhn-Tucker (KKT) conditions)** The following

four conditions are called KKT conditions (for a problem with differentiable f_i, g_i):

- Primal constraints: $f_i(x) \leq 0$, $i = 1, \ldots, I$, and $g_i(x) = 0$, $i = 1, \ldots, E$.
- Complementary slackness: $\beta_i f_i(x) = 0$, $i = 1, \ldots, I$.
- Dual constraints: $\beta \geq 0$.
- Gradient of Lagrangian with respect to x vanishes:

$$\nabla f_0(x) + \sum_{i=1}^{I} \beta_i \nabla f_i(x) + \sum_{i=1}^{E} \gamma_i \nabla g_i(x) = 0.$$

CHAPTER 3

BOUNDING OF K-CLUSTER PROBLEM AND SELECTED APPLICATIONS

3.1 Introduction

In this chapter, we begin by presenting the bound of the general formulation of K-CLUSTER PROBLEM, as well as reviewing some types of clusters and the most important applications that fall within this scope.

3.2 The General Formula of K-CLUSTER PROBLEM and Bound Procedure

The K-CLUSTER PROBLEM entails locating a subgraph with the heaviest weight and exactly k nodes ($1 < k < n - 1$) when there is an edge weighted graph with n vertices. This is a classic combinatorial optimization problem, also known as the (heaviest k-subgraph problem), (k-dispersion problem), (k-defence-sum problem), and (densest subgraph problem) when all edge weights are equal to 1. The k-cluster problem entails evaluating a subset $S \subseteq V$ of k vertices such that the number of the weights of the edges between vertices in S is maximized according to given a graph $G = (V, E)$. Letting $n = |V|$ denote the number of vertices, and w_{ij} denote the edge weight for $ij \in E$ and $w_{ij} = 0$ for $ij \notin E$ the problem can be modeled as the optimization problem:

$$\begin{cases} \text{Maximize} & \frac{1}{2} z^T W z \\ \text{subject to} & e^\top z = k \\ & z \in \{0,1\}^n, k \in \mathbb{Z}. \end{cases} \tag{3.1}$$

We rewrite this problem as follows

$$\begin{cases} \text{Maximize} & \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} z_i z_j \\ \text{subject to} & \sum_{i=1}^{n} z_i = k \\ & z \in \{0,1\}^n, k \in \mathbb{Z}. \end{cases} \tag{3.2}$$

With the vector of all ones $e = (1, \ldots, 1) \in \mathbb{R}^n$ and $W := (w_{ij})_{ij} \in \mathcal{S}^n$ is the weighted adjacency matrix of the graph G (which is a symmetric $n \times n$ matrix), [29, 66, 81, 82]. A feasible z (that is, such that $e^\top z = k$ and $z \in \{0,1\}^n$) also satisfies for all $j \in \{1, \ldots, n\}$

$$\sum_{i=1}^{n} z_i z_j = k z_j$$

The left-hand side of this equality is quadratic in z; so we introduce the symmetric $n \times n$-matrix C_j such that $\sum_{i=1}^{n} z_i z_j = \frac{1}{2} z^\top C_j z$. The natural inner product in matrix space $\mathcal{S}^n$ is defined for any $X, Y \in \mathcal{S}^n$ by

$$\langle X, Y \rangle = \sum_{i,j=1}^{n} X_{ij} Y_{ij} = \text{trace}(XY)$$

This inner product is very convenient for our purposes through the relation:

$$\forall z \in \mathbb{R}^n, \forall \quad W \in \mathcal{S}^n, \quad z^\top W z = \left\langle W, z z^\top \right\rangle .$$

That is the quadratic constraint is given by

$$z^\top W z = \langle W, Z \rangle.$$

For reformulation for (3.2), adding these n product constraints to come up with the following equivalent formulation of K-CLUSTER

$$\begin{cases} \max \quad \frac{1}{2} z^\top W z \\ e^\top z = k, \quad z \in \{0,1\}^n \\ z^\top C_j z = 2 k z_j, \quad j \in \{1, \ldots, n\} \end{cases} \tag{3.3}$$

It is generally known that when dualizing only the {0,1} constraints (rather than the linear

constraints) for a {0,1} quadratic problem, the best bound is found through Lagrangian duality (see e.g. [70]). Though this bound does not immediately lead to an SDP problem, it is analogous to the semidefinite relaxation of the general problem strengthened by product constraints as in (3.3) for k-cluster, [38]. Adding more redundant quadratic equality constraints will result in the same bound, so we'll remain with the current one in (3.3). We should also remember that replacing the n product constraints with a single constraint $\left(e^\top z - k\right)^2 = 0$ in (3.1) is an equivalent approach: the two formulations result in two SDP relaxations that produce the same bound. The main reason for preferring formula (3.3) for our developments is to make a numerical comparison between the two SDP formulas, see [81]. However, in order to provide different bounds that have an SDP-quality but that are less tight that (3.3) we will apply the SDP relaxation of the K-CLUSTER PROBLEM as the following form

$$\begin{cases} \max \quad \langle Q, X \rangle \\ \\ \langle Q_j, X \rangle = 4k - 2n, \quad j \in \{0, \ldots, n\} \\ \\ \langle E_i, X \rangle = 1, \quad i \in \{0, \ldots, n\} \\ \\ X \succeq 0. \end{cases} \tag{3.4}$$

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

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

The squared constraint $\left(e^\top z - k\right)^2 = 0$. For the real parameter is $\alpha \in \mathbb{R}$, we consider the

Lagrangian function

$$L(X;\alpha) := \langle Q, X\rangle - \frac{\alpha}{2}\left(\|X\|^2 - (n+1)^2\right)$$

and the associated dual function

$$\eta(\alpha) := \begin{cases} \min \quad L(X;\alpha) \\ \\ \langle Q_j, X\rangle = 4k - 2n, \quad j \in \{0,\ldots,n\} \\ \\ \langle E_i, X\rangle = 1, \quad i \in \{0,\ldots,n\} \\ \\ X \succeq 0. \end{cases} \tag{3.5}$$

By weak duality, each value $\eta(\alpha)$ is an upper bound for the optimal value of k-cluster, since we can write: for all feasible X for (3.3) and then in (3.5) ,

$$\langle Q, X\rangle = L(X,\alpha) \leq \eta(\alpha) \tag{3.6}$$

so that $\eta(\alpha)$ is upper bound for (3.3) indeed. Hence we have a new approach of SDP bounds $\eta(\alpha)$ (parameterized by $\alpha \in \mathbb{R}$) and this bound generalize the standard SDP bound (3.5). Therefore the following SDP problem gives a bound for the k-cluster problem:

$$(\mathrm{SDP}_I)\begin{cases} \text{maximize } \langle Q, X\rangle \\ \text{subject to } \langle Q_j, X\rangle = 4k - 2n \\ \mathrm{diag}(X) = e, X \succeq 0 \\ \langle E_i, X\rangle - 1 \end{cases} \tag{3.7}$$

For the sake of brevity, we will only describe the SDP relaxation (3.7) very generally; for more details about the formulation of this SDP relaxation we refer the interested reader to [81]. The vector of all ones is denoted by e (we let the dimension of e depend on the context), and we have

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

The best possible bound $F^\alpha(y, z)$ can be found by solving the problem

$$\begin{aligned} \text{minimize} \quad & F^\alpha(y, z) \\ \text{subject to} \quad & y \text{ free}, z \geq 0 \end{aligned}$$

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

Proposition 3.2.1. (Our semidefinite bounds for k-cluster). For $y \in \mathbb{R}^n$ and $z \in \mathbb{R}$, let the matrix $X(y,z) \in \mathbb{S}^n$ be defined by

$$X(y,z) := [Q - (dig(x) - e)]_+.$$

Let $\alpha > 0$ and let $F^\alpha : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ be the function defined by

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

Then, for all $y \in \mathbb{R}^n$ and $z \in \mathbb{R}_+$, we have that $F^\alpha(y,z)$ is a valid upper bound for problem (3.7); that is,

$$(\text{KC}) \leq F^\alpha(y,z), \quad \text{for all } y \in \mathbb{R}^n, z \in \mathbb{R}_+$$

The following proposition gives us an important fact that supports the practical use of these bounds and the above problem is a convex and differentiable problem.

Proposition 3.2.2. [65] (Differentiability). The function F^α is convex and differentiable; its gradients are given by

$$\nabla_y F^\alpha(y,z) = b - \frac{1}{\alpha} B\left(X_k(y,z)\right), \quad \text{and} \quad \nabla_z F^\alpha(y,z) = e + \frac{1}{\alpha} A_k\left(X_k(y,z)\right).$$

Proof. The function F^α can be interpreted as a dual function, which immediately implies its convexity. The differentiability of F^α follows from [83]. □

Algorithm 1: Improved Semidefinite Bounding for Augmented Lagrangian Method

1 Input: Scalars $\alpha_0 > 0, \epsilon_0 > 0$, and $0 < \alpha, \epsilon < 1$

2 Initial y and z variables $y_0 = 0 \in \mathbb{R}^n$ and $z_0 = \theta$.

3 for $k = 1, 2, \ldots$ do

4 $\left(y^k, \hat{z}^k\right) = \underset{y \in \mathbb{R}^n,}{\arg\min} \quad \underset{z \in \mathbb{R}_-}{\mathcal{L}^{\alpha_{k-1}}_{\mathcal{J}_{k-1}}} \left(y, z, x^{k-1}\right)$ by using a quasi-Newton method starting at $\left(y^{k-1}, z^{k-1}\right)$ such that

$$\max\left\{[Q - (dig(x) - e)]_+, [\langle E_i, X\rangle - 1]_\infty\right\} < \varepsilon_k, \quad \text{where} \quad X_k \leftarrow \frac{1}{\alpha_k} X_{I_k}\left(y_k, z_k\right)$$

5 Update the bound $F^k = \hat{F}^{\alpha^k}_{\mathcal{J}_{k-1}}\left(y^k, \hat{z}^k\right)$: where

$$F^\alpha(y, z) := \frac{1}{2\alpha}\left\|X_k(y, z)\right\|_F^2 + b^T y + e^T z + \frac{\alpha}{2}(n+1)^2.$$

.

3.3 Types of Clustering

We will deal with different types of graphs, depending on the types of edges that are used to connect pairs of vertices, a graph can either be directed or undirected. In fact, the word 'undirected' is, however, usually omitted when referring to an undirected graph. We want to further distinguish between graphs with and without multiple edges, since for our proposed methods only those graphs without multiple edges are of special interest. A formal differentiation between the types can be achieved by a proper analysis of the set of edges E , [43, 90]. This gives a detailed insight into the structure, because a graph is can be divided as follows:

1. Undirected without multiple edges.

2. With (combined) multiple edges.

3. Directed with (combined) multiple edges a multiset over the Cartesian product. In the following Figure , we depict some types for each of the mentioned graph categories, (see Figure 3.1):

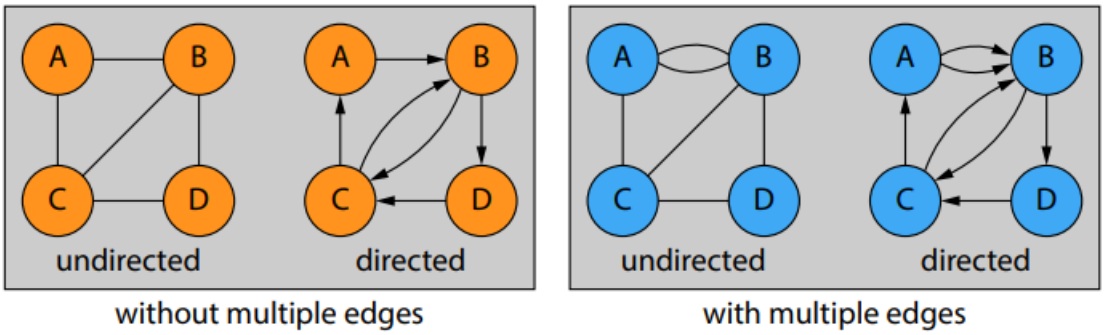

Figure 3.1: The vertices of those types that are of special interest to our proposed approaches are highlighted in orange color.

3.4 Problems on Clustering

Classical problem in graph theory is the computation of a path between two nodes. According to the definition, a path is a sequence of edges that connect a sequence of vertices such that all vertices are distinct from one another. Interestingly, while the shortest path between two nodes can be computed efficiently (e.g., with Dijkstra's shortest path algorithm), finding the longest path between two nodes is a challenging task for which no approach with polynomial runtime is known . Both problems are, however, indeed discrete since the problem formulation precisely asks to find a subset of edges that together fulfill a particular requirement. An edge is therefore completely inside or completely outside of a particular solution set, but cannot belong to the solution only fractionally. One could therefore formalize the problem by introducing a binary variable for each edge and define the solution to be represented by those variables that get a value of 1 assigned, while all other variables would get a value of 0 , [110, 119]. For example (see Figure 3.2):

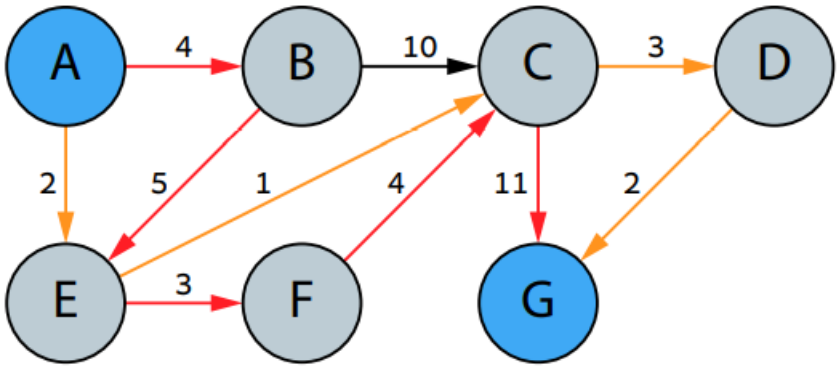

Figure 3.2: Shortest path (orange) of length 8 and longest path (red) of length 27 from node A to node G

3.5 Some Applications of Clustering Problem

The K-cluster problem is a fundamental graph optimization problem. A graph is an abstract concept for describing certain objects that are bound to one another in any way. The entity is referred to as a node, and the relationship between them is referred to as an edge. Graph partitioning is a method of distributing a disjoint subset of a graph's data to a separate unit. The need for sharing large graph data sets arises from the need to process data effectively and quickly in graph-related applications, [131]. Furthermore, the objective of clustering is to discover groupings (clusters) that are both homogenous and well separated; that is, components within a cluster should be similar, while elements in other clusters should be dissimilar. Clustering is the term for the process of creating clusters. The graph-theoretic technique to clustering involves creating a similarity graph from the data, with vertices corresponding to objects and edges connecting two vertices with similarity values greater than a certain threshold. According to the following principles, the information can offer more specific about the inner structure of the data set connected to the graph:

1. **Cliques** refers to the process of identifying subsets of nodes in which each pair of elements is linked.

2. **Clusters** refers to groupings of people who are highly linked.

3. Centrality (important nodes, hubs)

4. There are outliers.. (unimportant nodes)

The graph-based clustering relies on two main axes which is a collection of graph-theory-based clustering algorithms with a wide variety of applications and organize data in huge data sets to make it easier for consumers to get the information they need as shown in Figure (3.3).

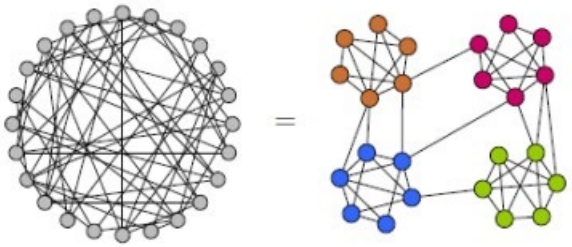

Figure 3.3: Clustering type

The idea of clustering structure is based on two concepts:

1. In a full or linked graph, objects are represented as nodes.

2. Give each branch between the two nodes x and y a weight. The distance $d(x, y)$ between the nodes determines the weight.

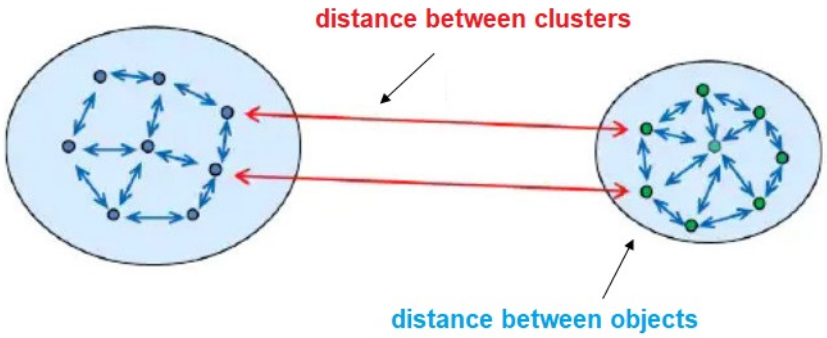

Figure 3.4: Clustering structure

Graph partitioning has been studied in the fields of computer science and artificial intelligence, [90] ,(see Figure 3.4). Finally, some applications for clustering problem will be reviewed in more detail in the following sections :

3.5.1 Applications of Clustering Problem in Social Networks

Sub-Graph pattern mining is a useful approach for clustering analysis and simplifying dense social networks. Number of linked components, network width, path length, number of neighbors, density, and other network characteristics are identified by Network Analyzer. Degree distribution, neighborhood connectivity,clustering coefficient, between centrality, closeness centrality, and other complicated characteristics are generated from the above. The method of finding similar terms used by people in social networks is known as textual similarity which provides data on terms that are often used by a group of individuals. clustering also is a branch of data mining [100]. Data mining is a crucial process that use a variety of approaches to extract patterns or information from large amounts of data. The social network is essentially a description of how individuals interact socially. People's social structure is defined by their shared relationship or interest. Simply put, social network analysis (SNA) is the study of social systems in order to comprehend their structure and function. It depicts and measures

the flow of information between individuals and groups. Twitter, Facebook, Google+, Instagram, Ibibo, and LinkedIn are some of the social media data sources. Written messages are primarily exchanged among community members on social networks.This process was done after applying the k-means clustering and spectral k-means clustering methods. The Spectral clustering technique uses data points are used as nodes in a linked network, and clusters are discovered by dividing the graph into subgraphs depending on its spectral decomposition. while K-means clustering means split the objects into k-clusters with the goal of minimizing a metric relative to the cluster centroids [112]. As shown in Figure (3.5) the cluster after and before applying k-means clustering and spectral k-means clustering . The graphs in A and C is before clustering for real dataset whereas the graphs in B after spectral k-means clustering for dummy dataset and in D the graph after spectral k-means clustering for Real dataset.

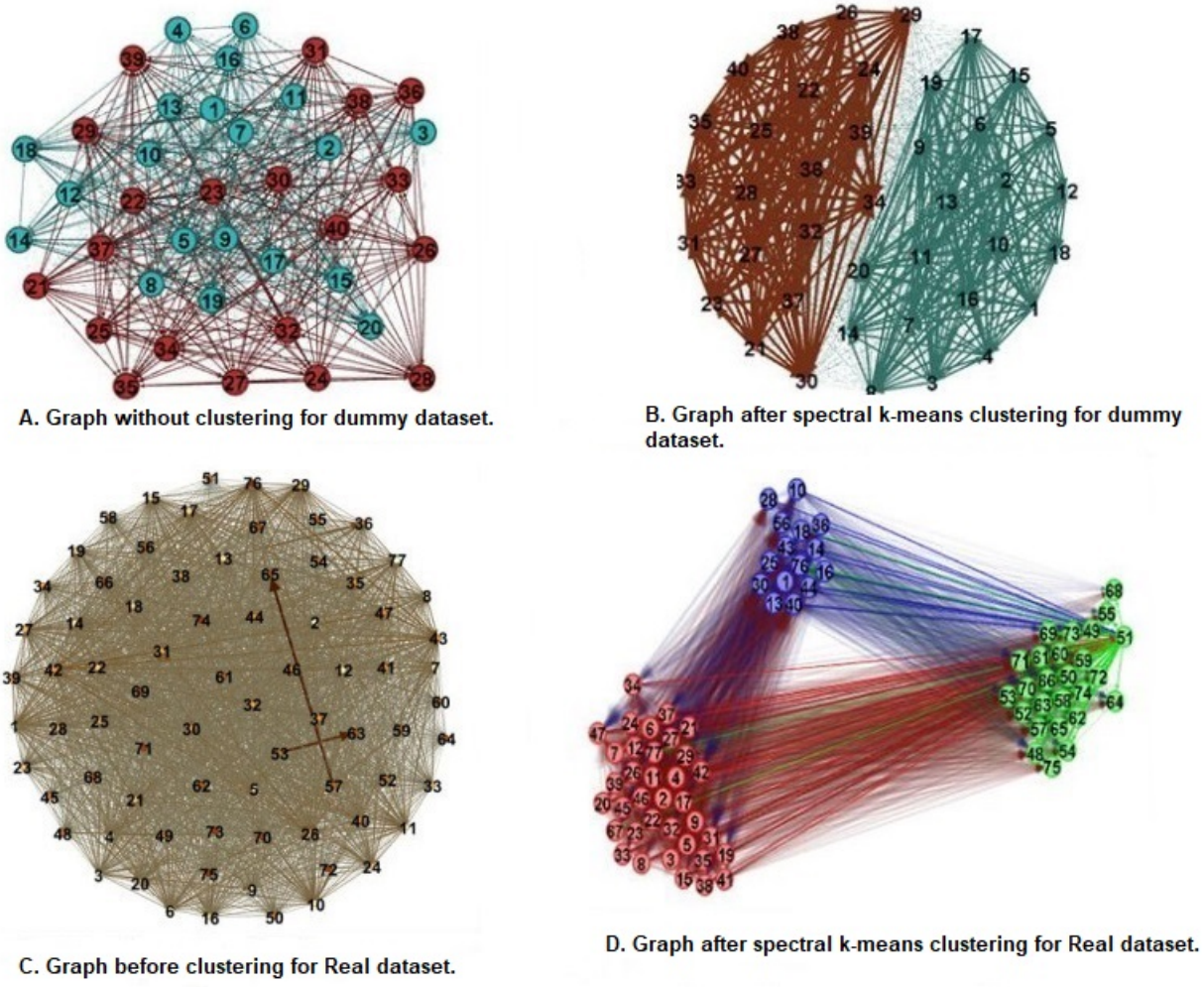

Figure 3.5: An example of a social network graph clustering technique.

3.5.2 Applications of Clustering Analysis in Real World

The objective of cluster analysis is to discover clusters in which the observations within each cluster are very similar to one another, but the observations in separate clusters are quite dissimilar and therefore in line with the principle of K-Clustering . The following examples demonstrate how cluster analysis is applied in a variety of real-world scenarios:

1. **Clustering in marketing :** Clustering is a technique used by retailers to discover groups of homes that are similar to one another. For example, a retailer may collect data on homes based on household income, household size, and head of household

occupation distance from the nearest metropolitan region, then feed the data into a clustering algorithm to see if the clusters can be identified and therefore based on how likely they are to respond to particular sorts of ads, the firm may then send tailored adverts or sales letters to each home, [51].Another example, customers are grouped based on their shopping habits based on an algorithm that is modeled according to the required data, [9].

2. **Clustering in streaming services :** Although data stream clustering has received a lot of attention in the last decade, there hasn't been much focus on dealing with streaming trajectories. Streaming services often use clustering analysis to identify viewers who have similar behavior. For example, a streaming service may collect the data about individuals in terms minutes watched per day, total viewing sessions per week, number of unique shows viewed per month and then using these metrics, a streaming service can perform cluster analysis to identify high usage and low usage users so that they can know who they should spend most of their advertising dollars on.Finally, the cluster plays a prominent role in online shopping ,many businesses use cluster analysis to identify consumers who are similar to each other so they can tailor their emails sent to consumers in such a way that maximizes their revenue, [84] .For example (see Figure 3.6):

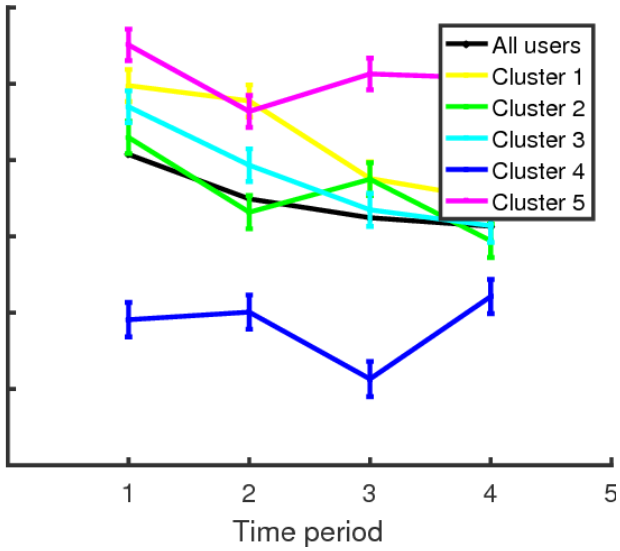

Figure 3.6: Clustering of streaming Services

3. **Clustering in the health field:** through the cluster structure, difficulties can be overcome. For example, within the current reality of the spread of the Corona virus, the most affected areas are identified and the situation is analyzed according to other categories in terms of the number of patients and the number of those who received the vaccine, in addition to identifying the most vulnerable people, identifying the affected places and drawing a map that is followed up periodically based on the data as clusters are configured accordingly. Most of the countries of the world used the cluster to limit the spread of the epidemic, as it was the only way to control the infection situation. The state of India had a very successful experience, and it reduced a lot of effort in the country through infection statistics and identification of infection clusters within regions, [67]. Finally, through the cluster structure, a map can be made showing vaccine areas, distribution centers, clinics, and hospitals, as shown in Figure (3.7):

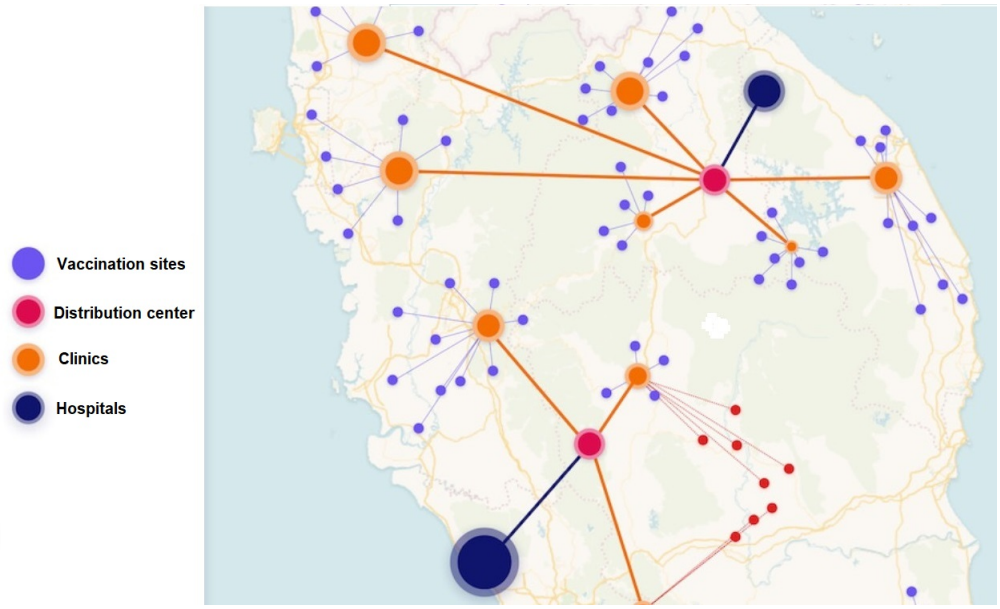

Figure 3.7: Clustering of health field .

3.5.3 Applications of Clustering Problem in Image Processing

Image segmentation is a central problem in any application's image processing. One of the most appealing methods for splitting an image into several components is graph partitioning. Pixels represent a vertex, and if two pixels are adjacent, they are represented as an edge, [42]. The image below (see Figure 3.8) has been classified according to the concept of clustering as follows:

1. In part(a), an image with the quadtree tessellation,
2. In part(b), the associated partition tree,
3. In part(c), a real image with the quadtree tessellation,
4. In part(d), the region adjacency graph associated to the quadtree partition,

While part (e) and (f) represent two different types of irregular tessellation of the image using image-dependent super-pixel segmentation methods by histogram cluster.

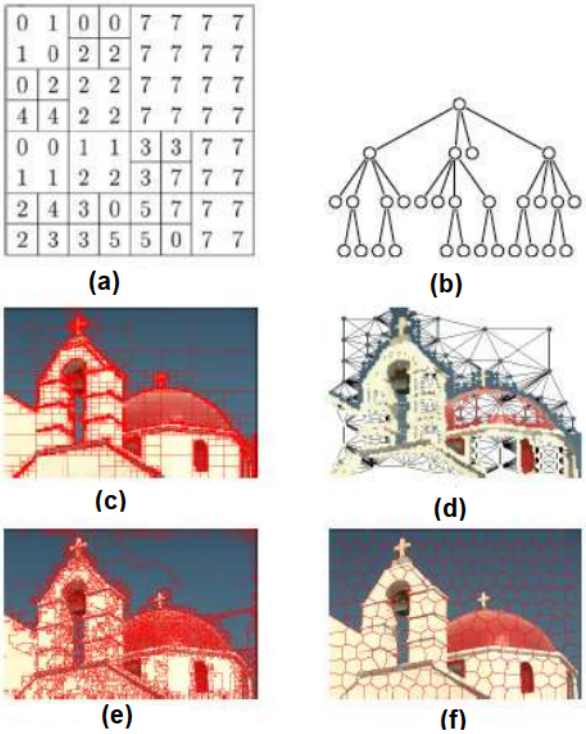

Figure 3.8: Image processing and graph partition

CHAPTER 4

APPROXIMATION METHODS

In this chapter, we begin by discussing the approximation methods and concepts that was used. Actually, the penalty method and the augmented Lagrangian method for both linear programming problem and semidefinite programming are explored in depth. Furthermore, approximation methods are a very active area in optimization. Consider the minimizing convex function $\mathcal{G}\colon \mathbb{R}^n \to \mathbb{R}$ on a convex set X. The goal of approximation methods is to replace $\mathcal{G}$ and X with approximation $\mathcal{G}^k$ and X^k. The approximation method works only if the approximation is easier than the original problem [12, 13]. For each iteration k we tried to find

$$X^{k+1} = \arg \min_{x \in X^k} \mathcal{G}^k(x),$$

then at the next iteration, $\mathcal{G}^{k+1}$ and X^{k+1} are generated by the approximation which depends on the new point x^{k+1} [12, 13]. Many great approximation methods are based on this idea, the penalty method, the augmented Lagrangian method will be adopted in the investigation axis to get the optimal solution [12, 13].

4.0.1 Quasi-Newton methods and Procedure

When complete Newton's Methods are either too time consuming or difficult to apply in Non-Linear Programming, Quasi-Newton Methods (QNMs) are used. These approaches are primarily used to obtain the global minimum of a twice-differentiable function f(x). For vast and complex non-linear problems, Quasi-Newton Methods have distinct advantages over the complete Newton's Method. However, depending on the sort of Quasi-Newton Method Uses and the situation to which it is applied, these methods are not ideal and can have significant limitations. The main advantage of QNMs is that they do not require iterative calculation of the inverse Hessian. As a result, the different types of QNMs are greatly reliant on the approximation utilized. For each iteration, the most basic approximation uses the same inverse Hessian value. As for the technique, it is very similar to the regular Newton method except for the modification of the Hessian Matrix step and

this depends on the type of method used for the Quasi-Newton method. Consider $f(a)$ that is twice-differentiable:

1. Choose a starting point a_o.
2. Calculate search direction by estimating H^{-1}.
3. Calculate change in a using the following. equation:

$$a^{k+1} = a^k - \left[H^{-1}\right]_k * \text{grad}\left(a^k\right)$$

4. Determine new a value, a^1.
5. By using a convergence criteria, determine if the technique has converged (gradient).
6. If the results aren't optimum, go back to step 2.

4.1 The Penalty and Augmented Lagrangian Methods

In an optimization problem, one wishes to maximize or minimize some function subject to some constraints. The general optimization problem given by [12, 13, 37]:

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to} \quad x \in X. \end{aligned} \tag{4.1}$$

The function f is defined from a convex set $X \subseteq \mathbb{R}^n$ into $\mathbb{R}$. A point $x^* \in X$ is a local solution of problem (4.1) if there exists a neighborhood $\mathcal{B}(x^*, t)$ such that $f(x^*) \leq f(x)$ for all $x \in \mathcal{B}(x^*, t) \cap X = \left\{x \in X \;\middle|\; \|x - x^*\| \leq t\right\}$.

4.1.1 Optimality Conditions for Unconstrained Optimization

In this section, we consider unconstrained optimization problem. If $X = \mathbb{R}^n$, i.e., minimize f without constraints [12, 13, 37], it can be expressed as:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x). \tag{4.2}$$

- If f is continuously differentiable, then a necessary condition for $x^* \in \mathbb{R}^n$ to be a solution of problem (4.2) is

$$\nabla f(x^*) = 0.$$

- If f is twice continuously differentiable, then a necessary conditions for $x^* \in \mathbb{R}^n$ to be a solution of problem (4.2) is

$$\nabla f(x^*) = 0, \;\; \nabla^2 f(x^*) \succeq 0.$$

- The sufficient conditions for $x^* \in \mathbb{R}^n$ to be a local solution of problem (4.2) are

$$\nabla f(x^*) = 0, \;\; \nabla^2 f(x^*) \succ 0.$$

Theorem 4.1.1. ***(First-Order Necessary Condition)*** *[13] Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be differentiable. If x^* is a local minimizer of f, then $\nabla f(x^*) = 0$.*

Theorem 4.1.2. ***(Second-Order Necessary Condition)*** *[13] Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be twice differentiable. If x^* is a local minimizer of f, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite.*

Theorem 4.1.3. ***(Second-Order Sufficient Condition)*** *[13] Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable. If $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite, then x^* is a strict local minimizer.*

4.1.2 Optimality Conditions for Constrained Optimization

The optimization problem is called the general nonlinear programming problem if we also have some equality (or inequality $h_j(x) \leq 0, \quad j = 1, \ldots, I$) constraints, or both of them, and it can be expressed as [12, 13, 37]:

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && g_i(x) = 0, \quad i = 1, \ldots, E, \\ & && x \in X, \end{aligned} \tag{4.3}$$

where f and g_i defined from $X \subseteq \mathbb{R}^n$ into $\mathbb{R}$ are assumed to be continuously differentiable functions. The feasible set of (4.3) is denoted by

$$S = \big\{ x \in X \mid g(x) = 0 \big\},$$

where g is the function with component functions $g_1, \ldots, g_E$. In our study we assume $X = \mathbb{R}^n$. The Lagrangian function $\mathcal{L}\colon \mathbb{R}^n \times \mathbb{R}^E \to \mathbb{R}$ is represented by

$$\mathcal{L}(x, \beta) = f(x) + \beta^T g(x),$$

where $\beta = (\beta_1, \ldots, \beta_E)^T$ is called the Lagrange multiplier vector. We have the following optimality conditions (see [12, 13, 37]).

Theorem 4.1.4. ***(First-Order Necessary Conditions)*** *[37] The first order necessary conditions (KKT conditions) for x^* to be a local minimum of problem* (4.3) *are that x^* must be a feasible point (i.e., $g(x^*) = 0$) and there exists a Lagrange multiplier vector β^* such that*

$$\nabla_x \mathcal{L}(x^*, \beta^*) = 0.$$

Theorem 4.1.5. ***(Second-Order Necessary Conditions)*** *[37] The second order necessary conditions for x^* to be a local minimum of problem* (4.3) *are that x^* must be a*

feasible point and

$$\nabla_x \mathcal{L}(x^*, \beta^*) = 0, \;\; u^T \nabla^2_{xx} \mathcal{L}(x^*, \beta^*) u \geq 0,$$

for all u that satisfies $\nabla g(x^*)^T u = 0$.

Theorem 4.1.6. ***(Second-Order Sufficient Conditions)*** *[37] The sufficient conditions for x^* to be a local minimum of problem (4.3) are that x^* is a KKT point (i.e., x^* and β^* satisfy the first order necessary conditions) and that*

$$u^T \nabla^2_{xx} \mathcal{L}(x^*, \beta^*) u > 0,$$

for every nonzero vector u that satisfies $\nabla g(x^*)^T u = 0$.

Theorem 4.1.7. *([13])* ***(Convergence of the augmented Lagrangian method)*** *Consider the inequality problem*

$$\begin{aligned} &\textit{minimize} && f(x) \\ &\textit{subject to} && h_j(x) \leq 0, \;\; j = 1, \ldots, m, \\ &&& x \in X. \end{aligned} \tag{4.4}$$

Let $h \colon \mathbb{R}^n \to \mathbb{R}^m$ be continuously differentiable and $\nabla h(x) = [\nabla h_1(x), \ldots, \nabla h_m(x)]$. Let the sequence x^k satisfy

$$\|\nabla_x \mathcal{L}_{\alpha^k}(x^k, \beta^k)\| \leq \epsilon^k,$$

for $k = 1, 2, \ldots$, where $\{\beta^k\}$ is bounded, and $\epsilon^k \to 0$, $\alpha^k \to 0$, $0 < \alpha^{k+1} < \alpha^k$, $\epsilon^k \geq 0$ for all k. Assume that a subsequence $\{x^k\}_N$ converges to a vector x^* such that $\nabla h(x^*)$ has rank m. Then

$$\left\{ [\beta^k + \frac{1}{\alpha^k} h(x^k)]_+ \right\} \to \beta^*,$$

where β^* is a vector satisfying, together with x^*, the KKT conditions

$$(KKT)\begin{cases}\nabla f(x^*) + \nabla h(x^*)\beta^* = 0, & \beta^* \geq 0,\\ \langle \beta^*, h(x^*)\rangle = 0, & h(x^*) \leq 0.\end{cases} \tag{4.5}$$

4.2 Theoretical Convergence Properties of The Penalty and Augmented Lagrangian Methods

In this section, we discuss the main existing convergence theorems. We begin with the convergence of the penalty and augmented Lagrangian methods for problem (4.3). Recall that

$$P_q(x, \alpha) = f(x) + \frac{1}{2\alpha}\|g(x)\|^2,$$

and, the augmented Lagrangian function is

$$\mathcal{L}(x, \beta, \alpha) = f(x) + \beta^T g(x) + \frac{1}{2\alpha}\|g(x)\|^2.$$

Let us define $D(x)$ to be $E \times n$ Jacobian of $g(x)$, where

$$g(x) = [g_1(x), \ldots, g_E(x)]^T.$$

Hence

$$D(x)^T = [\nabla g_1(x), \ldots, \nabla g_E(x)].$$

4.2.1 Theoretical Convergence Properties of The Penalty method

Theorem 4.2.1. ***(Convergence of penalty method)*** *[46] Let f and g be twice continuously differentiable functions of problem* (4.3)*. Let*

$$y^k = \frac{g(x^k)}{\alpha^k},$$

and

$$\|\nabla_x P_q(x^k, \alpha^k)\| \leq \epsilon^k,$$

where $\epsilon^k \to 0$ and $\alpha^k \to 0$ as $k \to \infty$. If x^k converges to x^*, where $\nabla g_i(x^*)$, $i = 1, \ldots, E$, are linearly independent, then x^* satisfies the first-order necessary optimality condition (KKT point) of problem (4.3) *and y^k converges to y^*, where y^* is the vector of Lagrange multipliers.*

Theorem 4.2.2. *[77]*
Let $\{x_k\}$ be a sequence generated by the penalty method. Then, any limit point of the sequence is a solution to (GP)

4.2.2 Theoretical Convergence Properties of The Augmented Lagrangian Methods

Theorem 4.2.3. ***(Convergence of augmented Lagrangian method)*** *[46] Let f and g be twice continuously differentiable functions. Let*

$$y^k = \beta^k + \frac{g(x^k)}{\alpha^k},$$

and

$$\|\nabla_x \mathcal{L}(x^k, \beta^k, \alpha^k)\| \leq \epsilon^k,$$

where $\epsilon^k \to 0$ *as* $k \to \infty$. *If* x^k *converges to* x^*, *where* $\nabla g_i(x^*)$, $i = 1, \ldots, E$, *are linearly independent, then* $y^k \to y^*$ *with* y^* *satisfying* $\nabla f(x^*) = D(x^*)^T y^*$. *If additionally, either* $\alpha^k \to 0$ *with bounded* β^k *or* $\beta^k \to y^*$ *with bounded* α^k, *then* x^* *satisfies the first-order necessary optimality condition (KKT point) of problem* (4.3) *and* y^* *is the vector of Lagrange multipliers.*

Proof: Since the augmented Lagrangian function is

$$\mathcal{L}(x, \beta, \alpha) = f(x) + \beta^T g(x) + \frac{1}{2\alpha} g(x)^T g(x),$$

we have that

$$\nabla_x \mathcal{L}(x, \beta, \alpha) = \nabla f(x) + D(x)^T \beta + \frac{1}{\alpha} D(x)^T g(x),$$

which can be rewritten as

$$\nabla_x \mathcal{L}(x, \beta, \alpha) = \nabla f(x) + D(x)^T \left(\beta + \frac{g(x)}{\alpha} \right).$$

Therefore, we have that

$$\|\nabla_x \mathcal{L}(x^k, \beta^k, \alpha^k)\| = \|\nabla f(x^k) + D(x^k)^T y^k\| \leq \epsilon^k,$$

by assumption. Then, following the proof of Theorem 4.2.1, we conclude that $y^k \to y^*$, where $y^* = -(D(x^*)^+)^T \nabla f(x^*)$, and that $\nabla f(x^*) + D(x^*)^T y^* = 0$. Now, by definition of y^k,

$$\|g(x^k)\| = \alpha^k \|\beta^k - y^k\| \leq \alpha^k \|y^k - y^*\| + \alpha^k \|\beta^k - y^*\|.$$

By assumption, $\alpha^k \to 0$ with bounded β^k or $\beta^k \to y^*$ with bounded α^k, so we have that $g(x^k) \to 0$ in either case. Since x^k converges to x^* and g is continuous, $g(x^*) = 0$. Thus (x^*, y^*) satisfies the first-order optimality conditions. ■

4.2.3 The Penalty Function Methods

A penalty method is an approach that uses a sequence of unconstrained problems. This method replaces a constrained problem with unconstrained problems by adding a penalty term to the objective function . In 1943, the first penalty function was studied by Courant [12, 13, 37]. However, sequential unconstrained minimization approaches began to be widely used for solving optimization problems in the 1960s (see [39]), and such approaches were applied to get the solution of the original constrained problems. Following [37], we consider the standard penalty function (quadratic penalty function) of problem (4.3) given by

$$P_q(x, \alpha) = f(x) + \frac{1}{2\alpha}\|g(x)\|^2, \tag{4.6}$$

where $\alpha > 0$. Let $x(\alpha)$ be a minimizer of $P_q(x, \alpha)$ for $\alpha > 0$. The penalty methods produce a sequence of infeasible solutions. Each iterate $x(\alpha^k)$ is either necessarily infeasible or a local optimal solution of problem (4.3). These methods are beneficial because of their comparative simplicity as they can use powerful methods for solving unconstrained problems. Although they have a solid theoretical background, they are not efficient in practice since the sequence of unconstrained problems does not produce an exact solution [37]. The penalty parameter of the penalty function method must go to zero as you can see in Figure(4.1) (for more details, see [12, 13, 37]). The penalty function method is summarized in Algorithm(2).

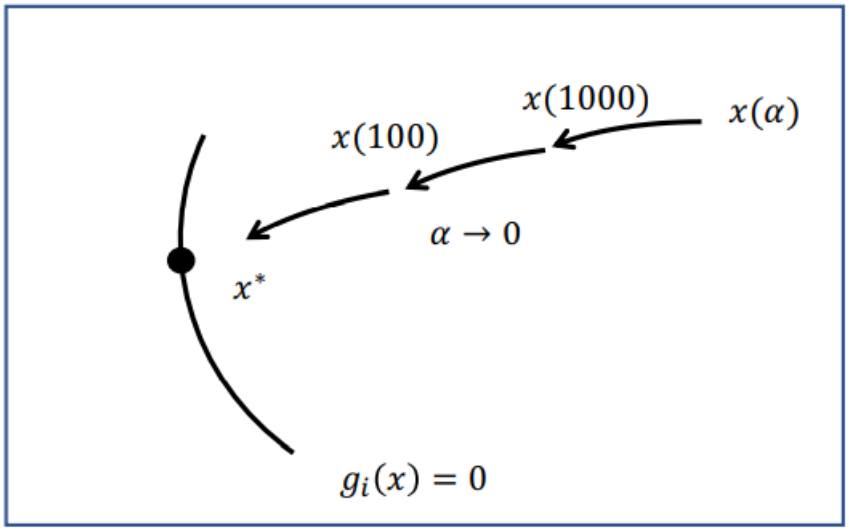

Figure 4.1: In penalty methods, $x(\alpha)$ approaches x^* as $\alpha \to 0$.

Algorithm 2: The Penalty Function Method

1 Given x^0, and $\alpha^0 > 0$.

2 Find x^{k+1} such that

$$x^{k+1} = \underset{x}{\operatorname{argmin}}\ P_q(x, \alpha^k).$$

3 Choose

$$\alpha^{k+1} \leq \alpha^k.$$

4 Set $k = k + 1$ and repeat.

4.2.4 The Augmented Lagrangian Method (Multiplier Methods)

This method began to be used in the 1970s. Initially, it was called the method of multipliers. Now, this method is known as the augmented Lagrangian method. The objective of this method is to solve constrained optimization problems. This is done by replacing a constrained problem with a sequence of unconstrained problems [12, 13, 37]. The augmented Lagrangian method is similar to the penalty method since in both of them a penalty term is added to the objective. The only difference in the augmented Lagrangian method is that a Lagrange multiplier term is added to it [12, 13, 37].

The augmented Lagrangian method was introduced by Hestenes [50]. To introduce the augmented Lagrangian method, we can change the constraint $g_i(x) = 0$ of problem (4.3) to the constraint $g_i(x) + \alpha\beta = 0$, as shown in Figure (4.2). Thus, we obtain the problem

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && g_i(x) + \alpha\beta = 0, \quad i = 1, \ldots, E, \\ &&& x \in X. \end{aligned} \tag{4.7}$$

We apply the penalty method to problem (4.7), then we can obtain the augmented Lagrangian function as follows. We begin with the penalty problem for problem (4.7)

$$\underset{x}{\operatorname{argmin}}\ f(x) + \frac{1}{2\alpha}\Big(g(x) + \alpha\beta\Big)^T\Big(g(x) + \alpha\beta\Big),$$

which expands to

$$\underset{x}{\operatorname{argmin}}\ f(x) + \frac{1}{2\alpha}\Big(g(x)^T g(x) + 2\alpha\beta^T g(x) + \alpha^2\beta^T\beta\Big),$$

and simplifies to

$$\underset{x}{\operatorname{argmin}}\ f(x) + \beta^T g(x) + \frac{1}{2\alpha}\|g(x)\|^2.$$

Therefore, the augmented Lagrangian function is

$$\mathcal{L}(x, \beta, \alpha) = f(x) + \beta^T g(x) + \frac{1}{2\alpha}\|g(x)\|^2. \tag{4.8}$$

The multiplier method consists of updating an estimate of the Lagrange multiplier β and sometimes the penalty parameter α at each iteration [37]. The multiplier method is summarized in Algorithm 3.

Algorithm 3: The Augmented Lagrangian Methods

1 Choose x^0, and $\alpha^0 > 0$, choose β^0.
2 Find x^{k+1} such that

$$x^{k+1} = \underset{x}{\operatorname{argmin}}\ \mathcal{L}(x, \beta^k, \alpha^k).$$

3 Update α^k and β^k.
4 Set $k = k + 1$ and repeat.

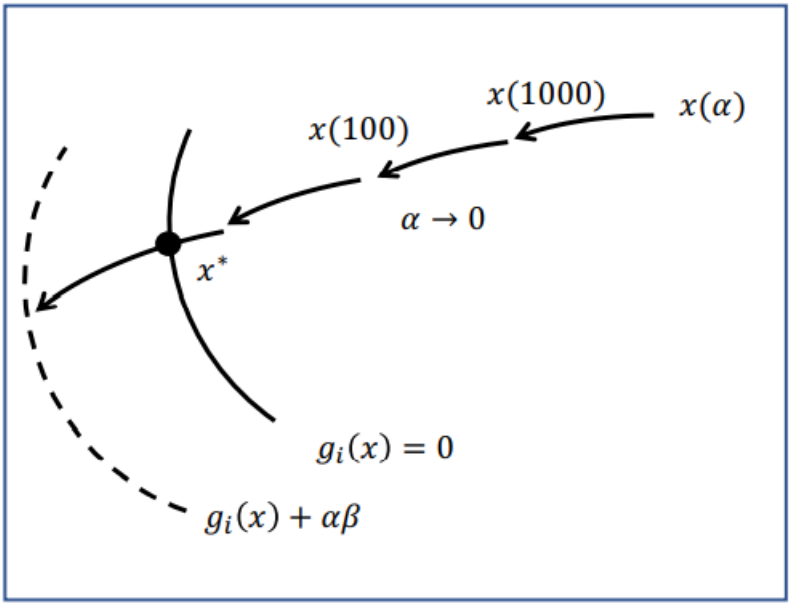

Figure 4.2: In the augmented Lagrangian method, we change $g_i(x) = 0$ to $g_i(x) + \alpha\beta = 0$, so we can attain x^* with a finite value of α.

As is mentioned in [102], Hestenes (1969) [50] and Powell (1969) [97] independently studied the formula for the estimate of the multiplier [37]

$$\beta^{k+1} = \beta^k + \frac{1}{\alpha^k} g(x^{k+1}). \tag{4.9}$$

In addition, Haarhoff and Buys [49], Buys (1972), Miele (1972), Miele et al. (1971, 1972), and Tapia (1974) presented different formulas for updating the Lagrange multiplier. The multiplier method that we used here is based on the multiplier update formula (4.9).

4.2.5 Relaxation of Approximate Methods for Solving NP-Hard Problems

It is well known that, the majority of real-world graph or network problems are NP-Hard which is hard to be solved. In this case, solving a simpler problem to gain estimates or constraints on the initial toughest challenge may be of interest, [1]. Consider the optimization problem where $f : R^n \to R$ and S subset from R^n

$$\begin{cases} \text{minimize } f(k) \\ \text{subject to } k \in S \end{cases}$$

The following form is a relaxation of the aforementioned problem:

$$\begin{cases} \text{minimize } f_R(k) \\ \text{subject to } k \in S_R \end{cases}$$

Where $f_R : R^n \to R$ such that $f_R(k) \leq f(k)$ for all k $\in S$ and $S \subseteq S_R$. It is obvious that the relaxation's optimum solution f_R^* is a lower constraint on the initial problem's optimal solution. According to the same concept, the Lagrangian relaxation aims to leverage the underlying network structure of these problems to implement these efficient methods. The Lagrangian Relaxation is a decomposition method: The problems constraints $S = S_1 \cup S_2$ are divided into two groups: the'easy' constraints S_1 and the'hard' constraints S_2. The hard restrictions are then eliminated, i.e., $S_R = S_1$, and the objective function, f_R relies on f and S_2, is transferred. The relaxation problem will be solved since S_R is a collection of easy' constraints. The Lagrangian relaxation is also interesting because, in some circumstances, the optimal solution of the relaxed problem is the same as the optimal solution of the original problem. To explain the principle of relaxation, we review

this simplified form, Consider we have hard problem represented by the form and formula below:

$$P_1 : \min_{k \in K} f(k)$$

When we make relaxation for this problem according to the principle of relaxation, we will get this form by which we can reach a quick and easy solution by applying theories and facts in this regard:

$$P_2 : \min_{k \in Y} f(k), K \subset Y$$

By comparing the diagram of the two functions (P_1 and P_2), we clearly find the concept of the following facts, see Figure (4.3).

$$(\text{ obj. of } P_1) \leq (\text{ Obj. of } P_2)$$

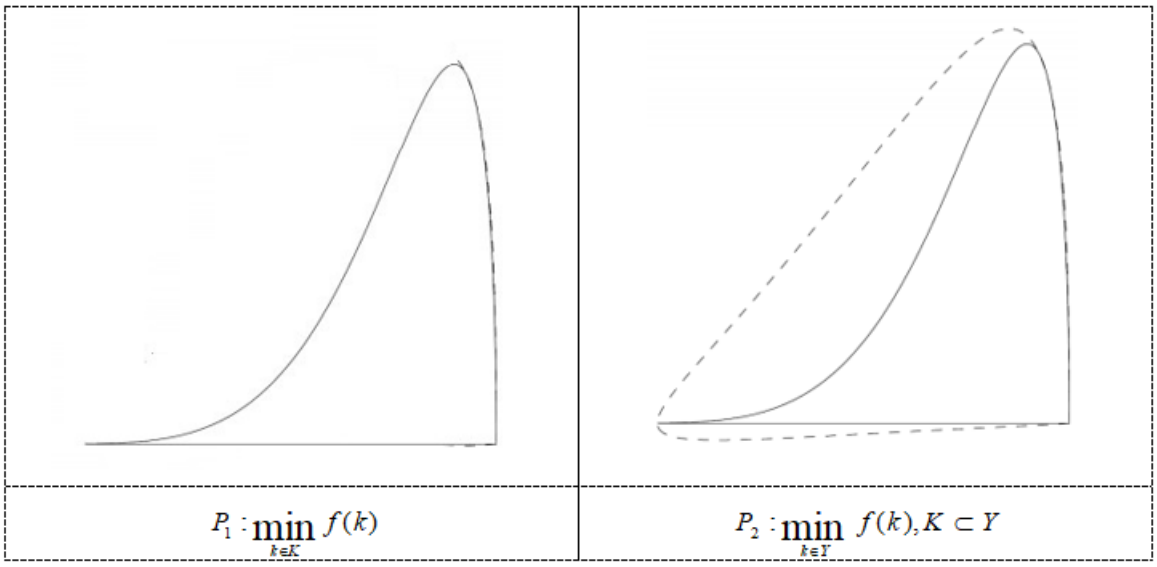

Figure 4.3: The problem after and before relaxation

proposition 1. If k^* is optimal for relaxation and feasible for exact, then k is optimal for exact.

Proof: assume k_l^* is relaxed optimal and feasible suboptimal for exact problem that is there exist y such that $f(y) < f(K_1^*), y \in K$. But according to the relaxation, $y \in Y$

and in this case k_l^* is not relaxed optimal, this leads to a contradiction!

4.3 Equality Augmented Lagrangian for Linear Programming

In this section, we present an approach known as the augmented Lagrangian method. We implemented the equality augmented Lagrangian. The initial results will provide an idea for working in semidefinite programming. [12, 13].

The generally LP problem is given as

$$(LP)\begin{cases}\text{minimize} & \langle c, x\rangle \\ \text{subject to} & \langle a_i, x\rangle = b_i, \quad i = 1, \ldots, m, \\ & x \geq 0, \quad x \in \mathbb{R}^n.\end{cases} \tag{4.10}$$

where $a_i \in \mathbb{R}^n$, for $i = 1, \ldots, m$, the numbers $b_i \in \mathbb{R}$, for $i = 1, \ldots, n$, and the vector $c \in \mathbb{R}^n$, $x_i \geq 0$, for $i = 1, \ldots, n$. The Lagrangian is given by

$$\mathcal{L}(x, y) = \langle c, x\rangle + \langle y, b_i - \langle a_i, x\rangle\rangle$$

The general form for the **augmented Lagrangian** is

$$\mathcal{L}_\alpha(x, y) = f(x) + \langle y, b_i - \langle a_i, x\rangle\rangle + \frac{1}{2\alpha}\|b_i - \langle a_i x\rangle\|^2. \tag{4.11}$$

Consider the primal (P) and the dual (D) standard linear programming problem:

$$(P)\begin{cases}\text{maximize} & \langle c, x\rangle \\ \text{subject to} & Ax = b, \\ & x \geq 0,\end{cases} \tag{4.12}$$

and

$$(D)\begin{cases}\text{minimize} & \langle b, y\rangle \\ \text{subject to} & A^T y \geq c\end{cases} \tag{4.13}$$

4.3.1 Summary

Here we provide a useful summary of the methods discussed above.

Problem	Penalty	Augmented Lagrangian
$\min \quad f(x)$ $\text{s.t.} \quad g(x) = 0$	$\min_x f(x) + \frac{1}{2\alpha}\|g(x)\|^2$	$\min_x f(x) + \beta^T g(x) + \frac{1}{2\alpha}\|g(x)\|^2$ $\beta^{k+1} = \beta^k + \frac{1}{\alpha^k} g(x^k)$
$\min \quad f(x)$ $\text{s.t.} \quad h(x) \leq 0$	$\min_x f(x) + \frac{1}{2\alpha}\|[h(x)]_+\|^2$	$\min_x f(x) + \frac{\alpha}{2}\left(\|[\beta + \frac{1}{\alpha}h(x)]_+\|^2 - \|\beta\|^2\right)$ $\beta^{k+1} = [\beta^k + \frac{1}{\alpha^k} h(x^k)]_+$

4.4 Bounding Procedure

Global optimality is typically hard to attain for NP-hard problems, necessitating the employment of other approaches. Typically, the goal is to discover a suboptimal solution that is close enough to the ideal solution quickly. It is critical to develop reasonable optimality bounds for the problem solution in this regard. We distinguish between primal and dual bounds, which offer upper and lower bounds for a minimization problem, respectively. Dual bounds need complicated relaxation techniques (e.g., linear, combinatorial, and Lagrangian) whereas primal bounds can be achieved by any feasible solution. According to this concept, and since we are focus to find optimal solution for K- cluster problem, the strategy will be to find the optimal value of the dual problem bounds of SDP, as shown in the following theorem:

Theorem 4.4.1. *[1] The optimal value of the dual problem bounds of SDP is the optimal value of* (KC) :

$$(KC) = \max_{X \in S_1} \left(1/2 z^T \mathrm{Wz}\right) \leq \max_{X \in S_2} \langle \mathrm{C}, X \rangle = (SDP)$$

Proof: Let the set $S_1 = \left\{ X \in S^n \mid \sum_{i=1}^{n} z_i = k, z \in \{0,1\}^n, k \in \mathbb{Z}. \right\}$ for the (KC) problem, and let the set $S_2 = \{ \quad \langle A_i, X \rangle = b_i \}$ for the SDP problem and then $S_1 \subseteq S_2$. Therefore,

$$(KC) \leq (SDP)$$

as desired. ■

To prove KC $\leq SDP < SPP \leq \theta(\gamma, z)$, where $\theta(\gamma, z) = \min \mathcal{L}_{\alpha d}(X, \gamma, S)$ of the duality of SDP. By theorem 4.4.1, we have $KC \leq SDP$ and we know that $SDP < SPP$, therefore

$$\mathrm{KC} \leq SDP < \theta(\gamma, z)$$

4.4.1 Function Call Algorithm

The number of function calls is computed by algorithms (augmented Lagrangian, and penalty methods) based on the bound and thus it is possible to determine which methods is faster to reach the bound .Where the method that accesses the SDP bound with the fewest number of function calls is the efficient method as we will see in the next chapter.

Algorithm 1 : Augmented Lagrangian (function call)

```
for i=1,...,n
    min (Augmented Lagrangian function call)
    for j=1,...,n
                Line search
        for k=1,...,n
            function call (How many )


        end
    end
end
```

Algorithm 2 : penalty method (function call)

```
for i=1,...,n
    min (penalty method function call)
    for j=1,...,n
                Line search
        for k=1,...,n
            function call (How many )


        end
    end
end
```

CHAPTER 5

NUMERICAL RESULTS

5.1 Introduction

In this chapter, we will discuss the numerical results for the algorithms that have been shown in previous chapters . In fact, the numerical tested were generated using the augmented Lagrangian method, which were confirmed by the Penalty method. Also, the results of a new hybrid algorithm between the Agumented Lagrangian method and the Penalty method were shown to be more accurace than the results of the two methods separately. Finally, there are detailed comparisons between the teste of the three numerical models.

5.2 Julia Language (JuliaBox)

Julia is a high-level, high-performance, dynamic programming language. While it is a general-purpose language and can be used to write any application.
Distinctive aspects of Julia's design include a type system with a parametric polymorphism in a dynamic programming language, with multiple dispatches as its core programming paradigm.
Julia was started in 2009, by Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman, who set out to create a free language that was both high-level and fast. On 14 February 2012, the team launched a website with a blog post explaining the language's mission [86].

The using Julia is by creating an account for the user and logging in to the site and working on it needs the availability of the internet. It is possible to work on it without the internet, but the condition that the packages that need constant updating are available.

5.3 Numerical Resultes

In this section, we'll go through our results and evaluate the performance of the suggested development method. These tested was carried out on a specific graph

imported from the Biq Mac library [125]. There are 100 nodes connected by 2475 edges, 80 nodes are connected by 1940 edges, and 50 nodes connected by 1214 edges in these graphs as shown in Figures (5.3, 5.2, 5.1) respectively, as well as on the other types of graphs. we implemented the Augmented Lagrangian, Penalty and Hybrid methods for solving the semidefinte programming. The figures in this section show the number of function calls made by the various techniques of solving K-CLUSTER PROBLEM. The exact solution to our problem was given by SB and CSDP solvers [21] . Types graphs of various sizes were tested, and the results were extracted and displayed in this part. Finally, our results depend on a new approach by using backtracking line search instead of huge line search.

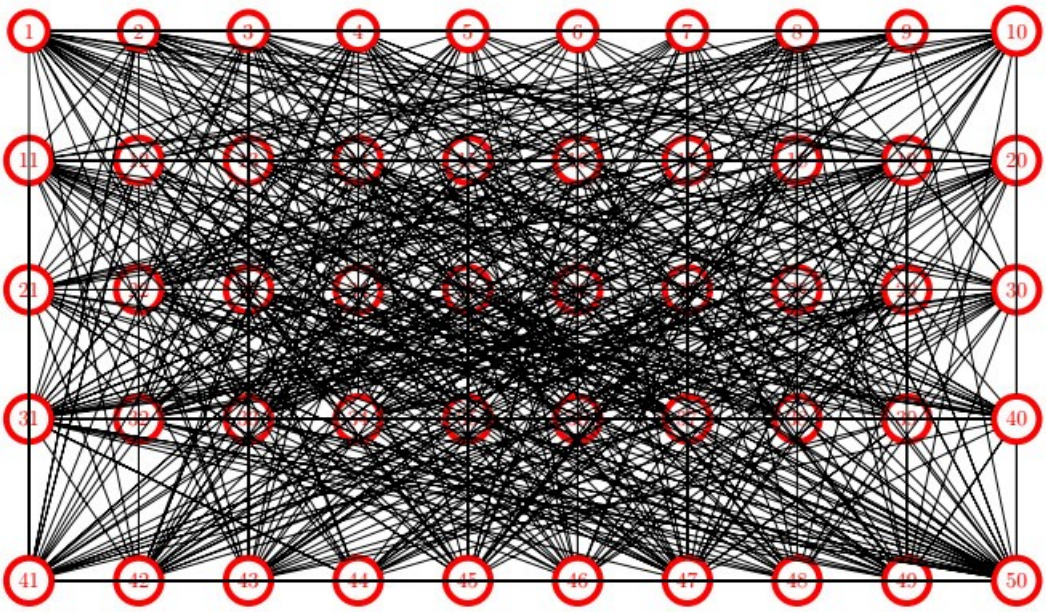

Figure 5.1: design the graph from Big Mac library and contain (50 vertices and 1214 edges)

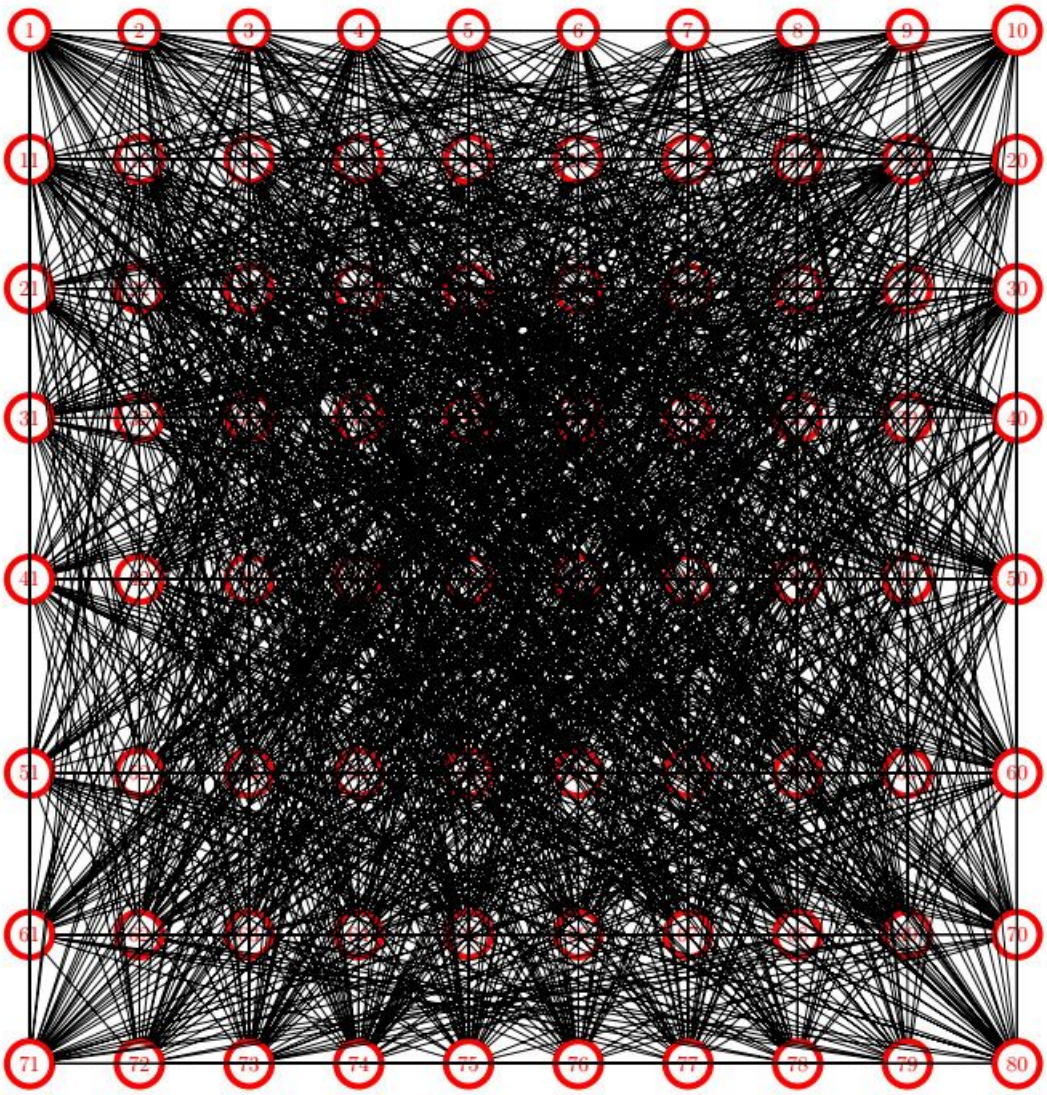

Figure 5.2: Design the graph from Big Mac library which contain (80 vertices and 1940 edges)

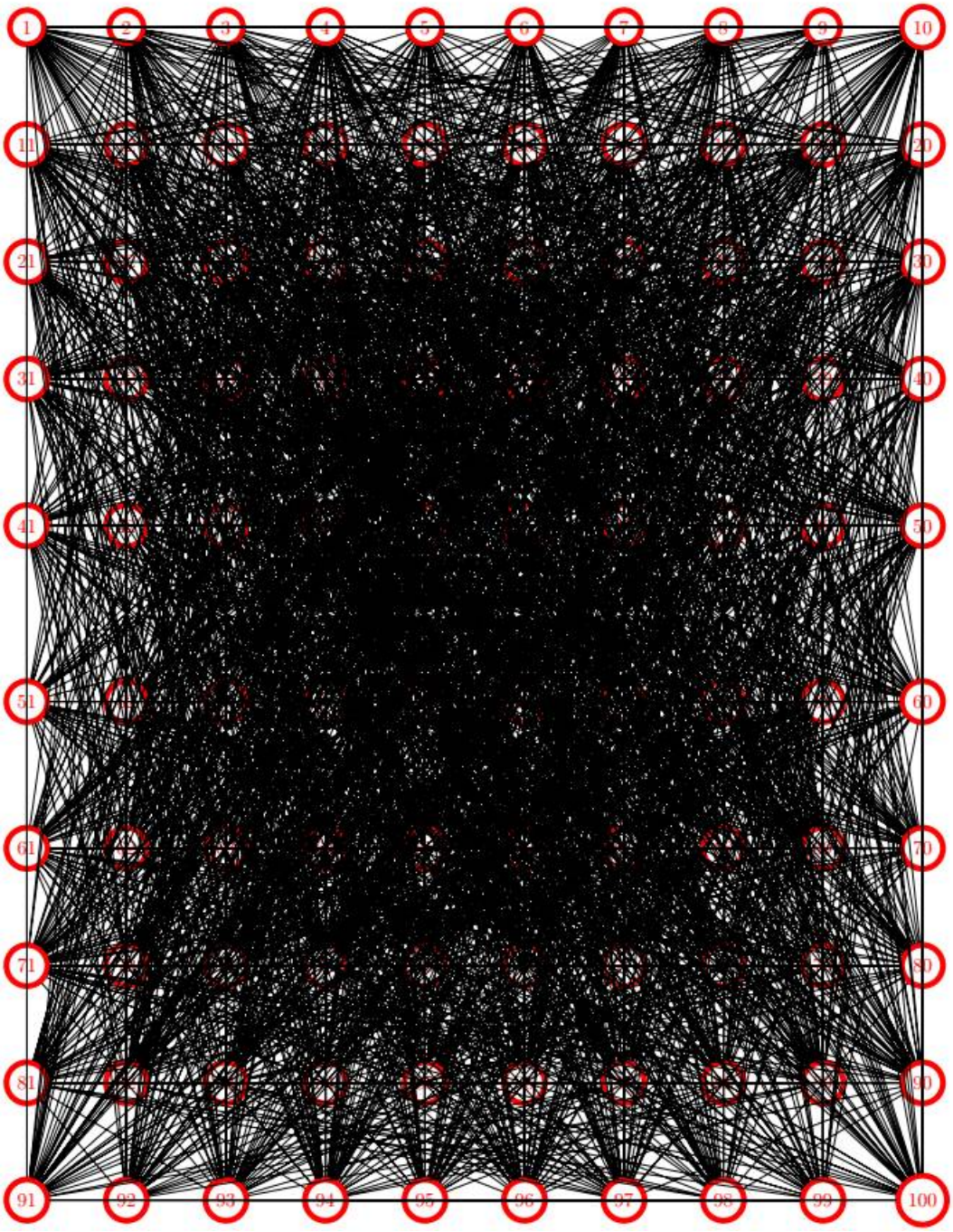

Figure 5.3: The graph select from Big Mac library and contain (100 vertices and 2475 edges)

5.3.1 Elementary Numerical Results of Graphs g05-60 (Function Calls)

In order to achieve the idea accurately and to obtain results through which it is possible to infer the fastest method of convergence to the bound of optimal solution, we've picked various problems (g05-60) from the Big Mac Library to work on as shown in Table (5.1) and the graphs in Figure (5.4). In fact, these problems contain 60 nods and 885 edges. Also, we used three approaches in these tests: Penalty method, Augumented Lagrangian method and Hybrid method. It was shown that the Augmented Lagrangian Method is more accurate than the method of Penalty Method, which means that the Augmented lagrangian method is the best, therefore it is provide the optimal solution of this cluster.

Problem	**PENfcalls**	**AUG.fcalls**	**Hybrid fcalls**
g05_60.0	754	588	**530**
g05_60.1	850	**444**	494
g05_60.2	410	253	**226**
g05_60.3	891	850	**524**
g05_60.4	490	**350**	466
g05_60.5	874	**322**	445
g05_60.6	504	**238**	269
g05_60.7	788	515	**480**
g05_60.8	446	230	**215**
g05_60.9	518	**271**	393
Total winner fcalls	**0**	**5**	**5**

Table 5.1: Numerical results of graphs (g05-60)

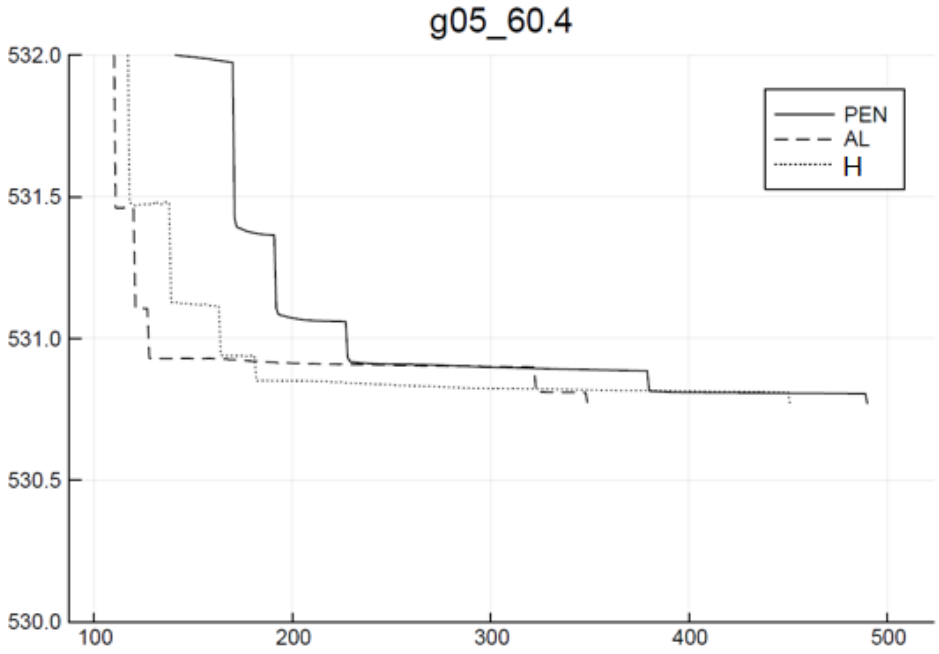

Figure 5.4: Convergence of the optimal solution among the methods

5.3.2 Elementary Numerical Results of Graphs g05-80 (Function Calls)

we applied the same previous technique, but to other samples of problems. As indicated in Table (5.2) and graph in Figure (5.5), we've chosen some problems (g05-80) from the Big Mac Library to work on . Actually, there are 80 nods and 1580 edges in these problems. In addition, these tests employed two approaches: penalty and Lagrangian methods. Furthermore, we compared the results and developed a new approach that hybrid the two methods and has shown promising results in terms of speed and accuracy. Also,tables (5.2) demonstrate that the Augmented lagrangian method is more accurate than the Penalty method, implying that the Augmented lagrangian method is the best, and also it is the best method to find the optimal solution to this cluster.

Problem	PENfcalls	AUG.fcalls	Hybrid fcalls
g05_80.0	476	**254**	391
g05_80.1	1043	499	**381**
g05_80.2	713	**285**	325
g05_80.3	320	**157**	173
g05_80.4	680	273	**269**
g05_80.5	423	**192**	269
g05_80.6	459	256	**238**
g05_80.7	413	**210**	254
g05_80.8	445	213	**190**
g05_80.9	337	215	**187**
Total winner fcalls	**0**	**5**	**5**

Table 5.2: Numerical results of graphs (g05-80)

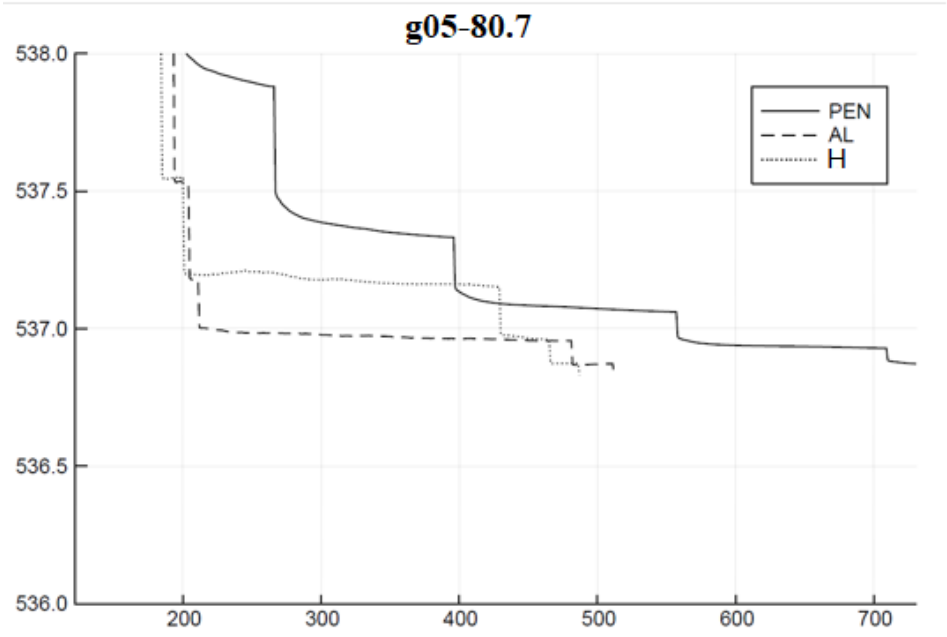

Figure 5.5: Convergence of the optimal solutions among the methods

5.3.3 Elementary Numerical Results of Graphs g05-100 (Function Calls)

The same approach was also applied to the previous two types of cluster problems, but on different style of graphs. We've picked some problems (g05-100.8) from the Big Mac Library to work on, as shown in Figure (5.6). These problems has a total of 100 nods and 2475 edges. In addition, we used two methodes in this test: Penalty and Augmented lagrangian method. Besides, we compared the results and devised a new approach that hybrid the two methods and has showed promising results in terms of speed and accuracy. Finally, the augmented lagrangian method is more accurate than the Penalty method depend on theoretical convergence properties, suggesting that the Augment method is superior to the Penalty method, see Table (5.3).

Problem	**PENfcalls**	**AUG.fcalls**	**Hybrid**
g05_100.0	401	**156**	232
g05_100.1	261	204	**188**
g05_100.2	517	240	**228**
g05_100.3	296	182	**163**
g05_100.4	509	**266**	284
g05_100.5	552	**221**	226
g05_100.6	484	320	**255**
g05_100.7	525	371	**253**
g05_100.8	438	**200**	314
g05_100.9	296	279	**195**
Total winner fcalls	**0**	**4**	**6**

Table 5.3: Numerical results of graphs (g05-100)

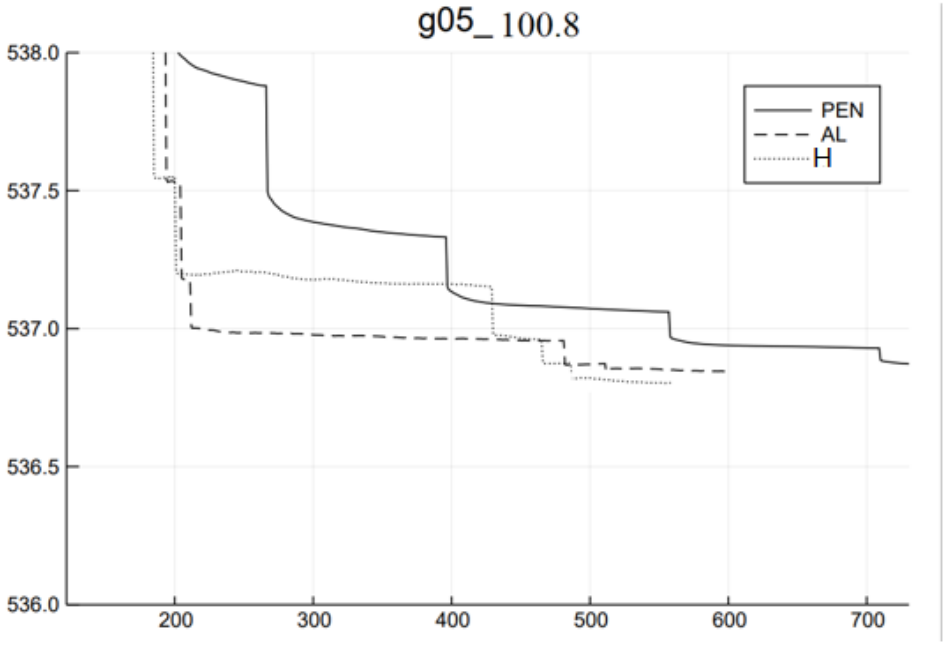

Figure 5.6: Convergence of the optimal solution among the methods

5.3.4 Numerical Results of Graphs(w01-100, w05-100 and w09-100)

As shown in the Figures ((5.7),(5.8) and (5.9))the results of evaluating several techniques on a graph with 100 nodes and 2475 edges in three models which is (w01-100) ,(w05-100) and (w09-100). In deed, we compared the outcomes of three Approximate methods (penalty method, augmented Lagrangian method, and a hybrid method). The hybrid technique was devised and used to solve the problems, which combines the Penalty method with the Augmented Lagrangian method. When both methods are used independently, this new approach algorithm appears to perform better. Also, according to the three models mentioned earlier, we show three tables of results as well as three figures of the progress of the solution process. Besides, the figures above indicate the SDP solution is equal to (602), (1660) and (2880) whereas the bound of semidefinite is equal to (632.5), (1667) and (2887.5) respectively. The hybrid approach gave a faster convergence to the optimum semidefinite bound, as seen in these Figures. The hybrid technique approaches

the semidefinite bound, as seen as approximately (483), (391) and (598) function calls, while the augmented Lagrangian method approaches the semidefinite bound as seen as approximately (394),(461) and (376) function calls, and the penalty method approaches the semidefinite bound late as usual ,as seen as approximately (1780),(622) and (1039) function calls. Finally, Tables ((5.4), (5.5) and (5.6)) shown the results out in detail, where reaching the bound with the fewest number of function calls is to save time and thus judging the method is the fastest.

$n = 100$, $density = 0.1$					
Graphs	**Optimal value**	**Penalty**	**Augment-ed**	**Hybrid**	**The goal bound**
w01_100.0	602	1780	**394**	483	632.45771
w01_100.1	719	1618	**1207**	1440	719.57932
w01_100.2	676	912	**663**	847	692.56110
w01_100.3	813	**904**	1301	1895	814.92739
w01_100.4	668	863	**791**	849	668.00791
w01_100.5	643	1695	**1202**	1375	645.17064
w01_100.6	654	1114	1484	**1106**	654.04204
w01_100.7	725	1695	1945	**1185**	728.28042
w01_100.8	721	840	**739**	808	721.00644
w01_100.9	729	1170	**855**	907	729.02160
Total winner fcalls		**1**	**7**	**2**	

Table 5.4: The test types of graphs (w01- 100) was selected from the Biq Mac library by the function calls of (augmented Lagrangian , penalty, hybrid) methods.

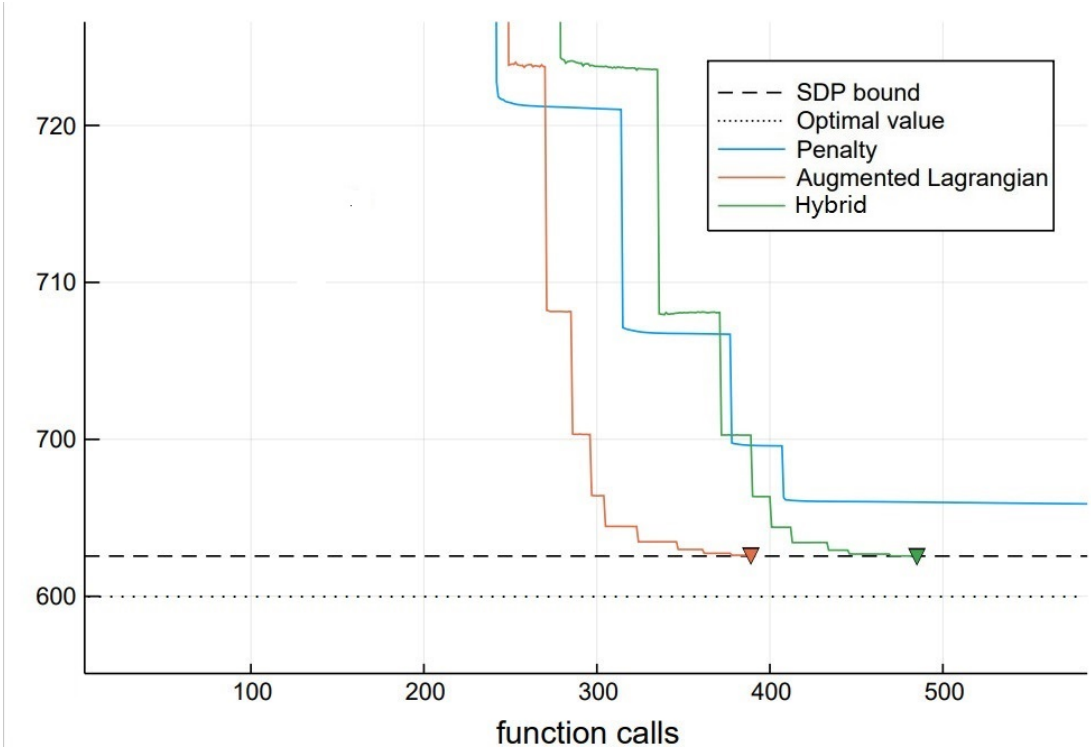

Figure 5.7: On graph (w01- 100.0), bounds versus function calls for Augmented Lagrangian, penalty methods, and hybrid technique where y-axis represented the bound.

$n = 100$, $density = 0.5$					
Graphs	**Optimal value**	**Penalty**	**Augment-ed**	**Hybrid**	**The goal bound**
w05_100.0	1646	641	535	**367**	1739.41303
w05_100.1	1606	636	512	**444**	1668.85603
w05_100.2	1902	895	**548**	694	1958.78729
w05_100.3	1627	731	820	**360**	1714.91680
w05_100.4	1546	**384**	450	467	1641.78351
w05_100.5	1661	622	461	**391**	1676.97781
w05_100.6	1479	968	**879**	907	1544.62023
w05_100.7	1987	907	**103**	445	2032.24223
w05_100.8	1311	681	736	**356**	1409.28063
w05_100.9	1752	758	993	**703**	1791.86734
Total winner fcalls		**1**	**3**	**6**	

Table 5.5: The test types of graphs (w05-100) was selected from the Biq Mac library by the function calls of (augmented Lagrangian, penalty, hybrid) methods.

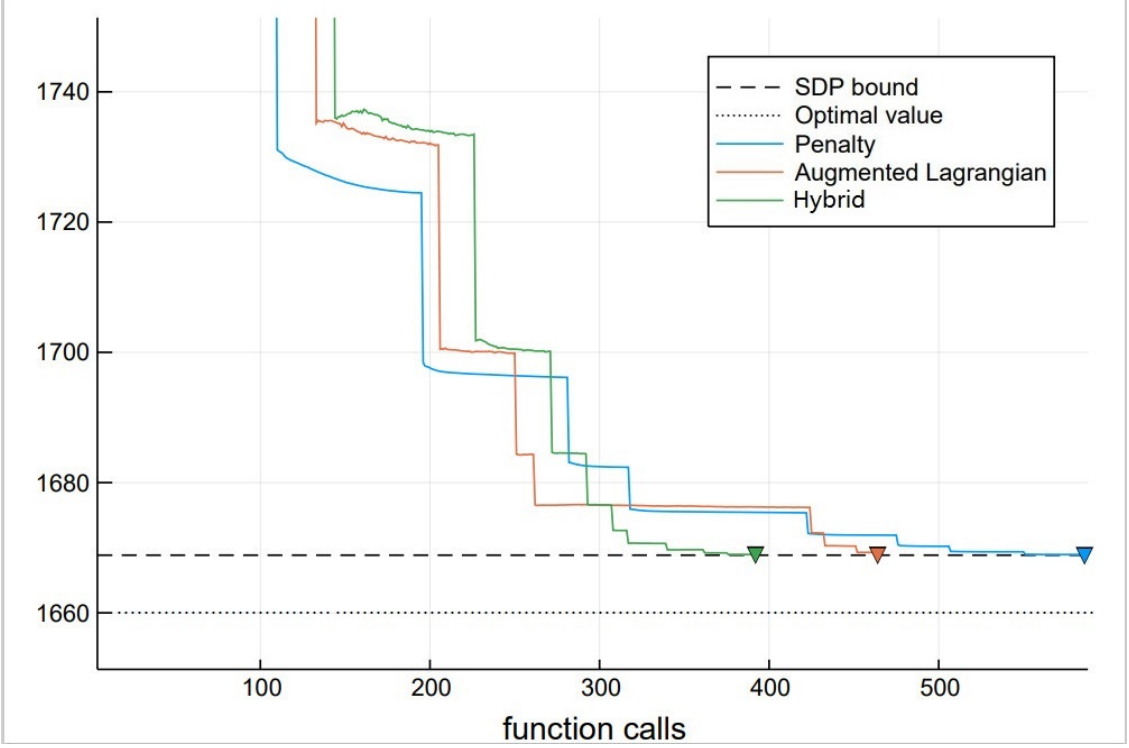

Figure 5.8: On graph (w05-100.5), bounds versus function calls for Augmented Lagrangian, penalty methods, and hybrid technique where y-axis represented the bound.

$n = 100$, $density = 0.9$					
Graphs	**Optimal value**	**Penalty**	**Augment-ed**	**Hybrid**	**The goal bound**
w09_100.0	2121	693	703	**524**	2235.05951
w09_100.1	2096	633	**575**	953	2264.55563
w09_100.2	2738	623	**426**	595	2881.52059
w09_100.3	1990	477	638	**316**	2133.05091
w09_100.4	2033	725	745	**542**	2155.59192
w09_100.5	2433	1364	**589**	618	2455.72668
w09_100.6	2220	909	**406**	549	2282.87760
w09_100.7	2252	653	**425**	512	2356.16199
w09_100.8	2876	1039	**376**	589	2925.04385
w09_100.9	2943	968	**438**	503	3162.28285
Total winner fcalls		**0**	**7**	**3**	

Table 5.6: The test types of graphs (w09-100) was selected from the Biq Mac library by the function calls of (augmented Lagrangian, penalty, hybrid) methods.

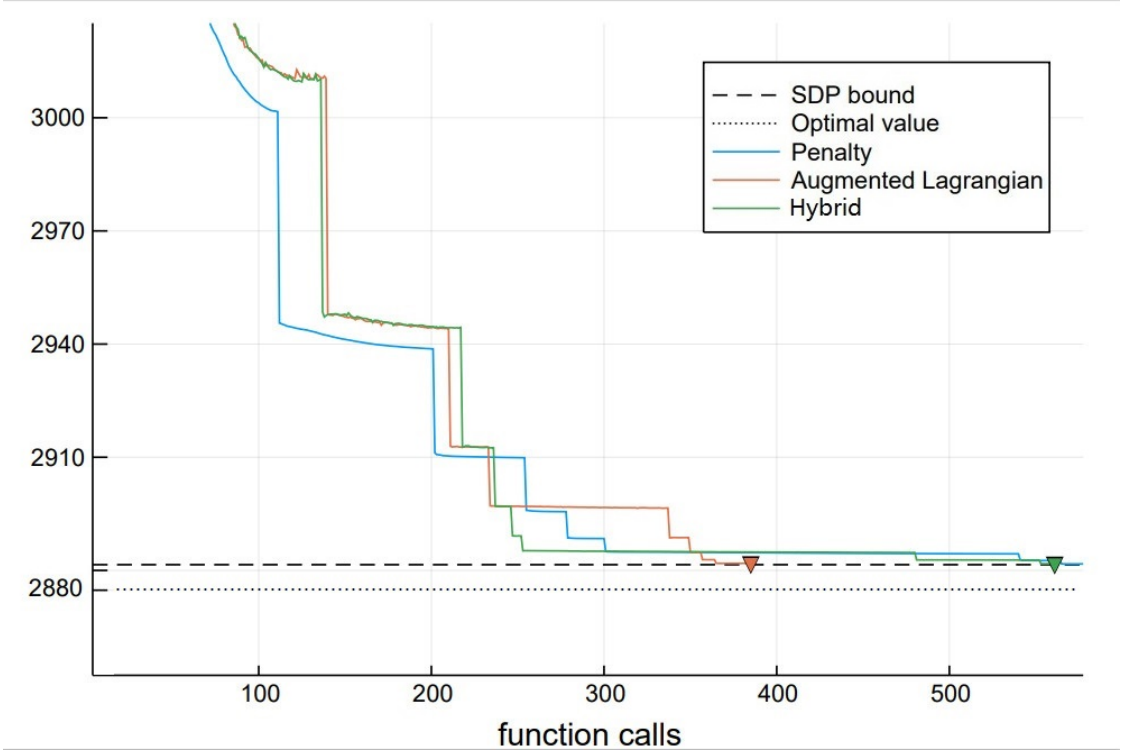

Figure 5.9: On graph (w09- 100.8) bounds versus function calls for Augmented Lagrangian, penalty methods, and hybrid technique where y-axis represented the bound.

5.3.5 Numerical Results of Graphs (be100.5)

In a related context, we will review the problem (be100.5) which was selected from the Biq Mac library. As shown in the Figure (5.10) of the type (be100.5) the SDP solution in this figure appears to be equal to 1586, while the K-CLUSTER bound is equal to 1591. Furthermore, by comparing the results of the three methods (penalty method, augmented Lagrangian method, and a hybrid method), we note that the chances of winning by approaching to semidefinite bound was for augmented Lagrangian method, where it reached the bound by 378 function calls , while the hybrid method had less luck than that, as it reached 477 function calls and then followed by the penalty method by 2003 function calls . The results for the problem (be100) are mentioned in Table (5.7) with meaning that getting to the bound with the fewest number of function calls is saving time, so the method is judged to be the fastest.

$n = 100$, $density = 0.1$					
Graphs	**Optimal value**	**Penalty**	**Augment-ed**	**Hybrid**	**The goal bound**
be100.1	19412	1599	**1499**	1511	19419.5
be100.2	17290	1780	**1630**	1733	17292.7
be100.3	17565	2680	**2000**	2510	17566
be100.4	19125	1670	2160	**1587**	19167.2
be100.5	15868	2003	**378**	477	15918.4
be100.6	17368	1605	1640	**1570**	17374.6
be100.7	18629	1840	2500	**1300**	18671.9
be100.8	18649	2140	1409	**1500**	18817.6
be100.9	13294	1540	**1502**	1925	13408.4
be100.10	15352	**1800**	1983	1980	15408.8
Total winner fcalls		**1**	**5**	**4**	

Table 5.7: The test types of graphs (be100)and (be200.8) was selected from the Biq Mac library by the function calls of (augmented Lagrangian , penalty, hybrid) methods.

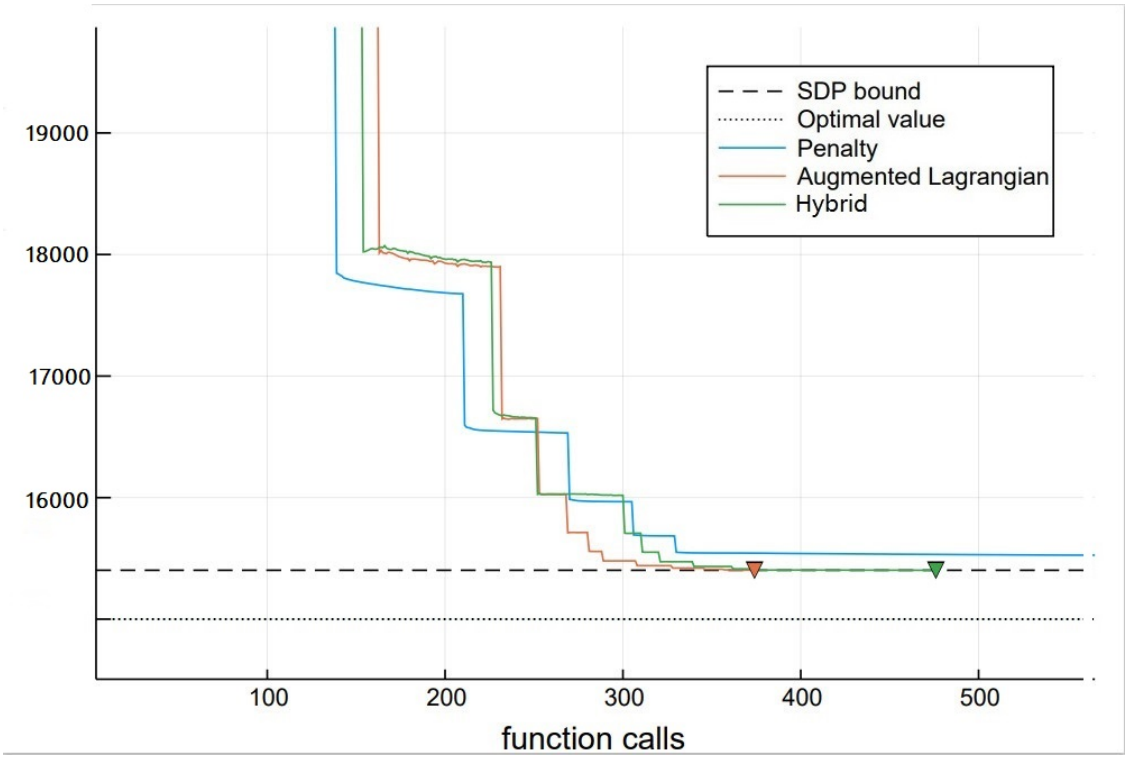

Figure 5.10: On graph (be100.5), bounds versus function calls for Augmented Lagrangian, penalty methods, and hybrid technique where y-axis represented the bound.

5.3.6 Numerical Results of Graphs (be120.3 and be120.8)

As shown in Table (5.8) and Figure (5.11) of graphs types (be120.3 and be120.8) depicts the convergence of a solution . Actually, the sample (be120.8.2) was the focus of the test, the hybrid method clearly has better convergence characteristics than the other two methods (penalty and augmented Lagrangian). The solution of SDP in this figure appears to be equal to 1882, while the K-CLUSTER bound is equal to 1902. The hybrid method algorithm attained this bound faster at 444 function calls while the Augmented Lagrangian method reach that by 449 function calls, but the penalty method reach that by 1802 function calls. In this graph, the penalty method gave a slower convergence time to attain the same bound. Getting to the bound with the fewest number of function calls is saving time, so the method is judged to be the fastest. Finally, hybrid approach is deemed to be the faster to reach the bound with the fewest amount of function calls.

Graphs	**Optimal value**	**Penalty**	**Augment-ed**	**Hybrid**	**The goal bound**
be120.3.1	13067	2200	2191	**2049**	13100.8
be120.3.2	13046	1200	1100	**1054**	13046.2
be120.3.3	12418	1173	1062	**1000**	12418.3
be120.3.4	13867	2922	2200	**1512**	13873
be120.3.5	11403	1589	**1009**	1321	11403
be120.3.6	12915	987	**900**	1804	12915
be120.3.7	14068	758	770	**749**	14068
be120.3.8	14701	**689**	748	825	14701
be120.3.9	10458	2002	1660	**1559**	10486
be120.3.10	12201	1520	1195	**1192**	12203.8
be120.8.1	18691	2321	2550	**2000**	18989.2
be120.8.2	18827	1802	449	**444**	19022.7
be120.8.3	19302	1846	**1408**	1533	19458.4
be120.8.4	20765	**1760**	1955	2473	20791
be120.8.5	20417	2090	2058	**2004**	20440.9
be120.8.6	18482	1789	**1270**	1510	18602.5
be120.8.7	22194	1900	2056	**1899**	22432.2
be120.8.8	19534	2077	2300	**2022**	19907.1
be120.8.9	18195	1983	**1095**	1890	18357.9
be120.8.10	19049	2230	**2080**	2381	19128
Total winner fcalls		**2**	**6**	**12**	

Table 5.8: The test types of graphs (be120.3 which has n=120 and d=0.3) and (be120.8 which has n=120 and d=0.8) was selected from the Biq Mac library by the function calls of (augmented Lagrangian , penalty, hybrid) methods.

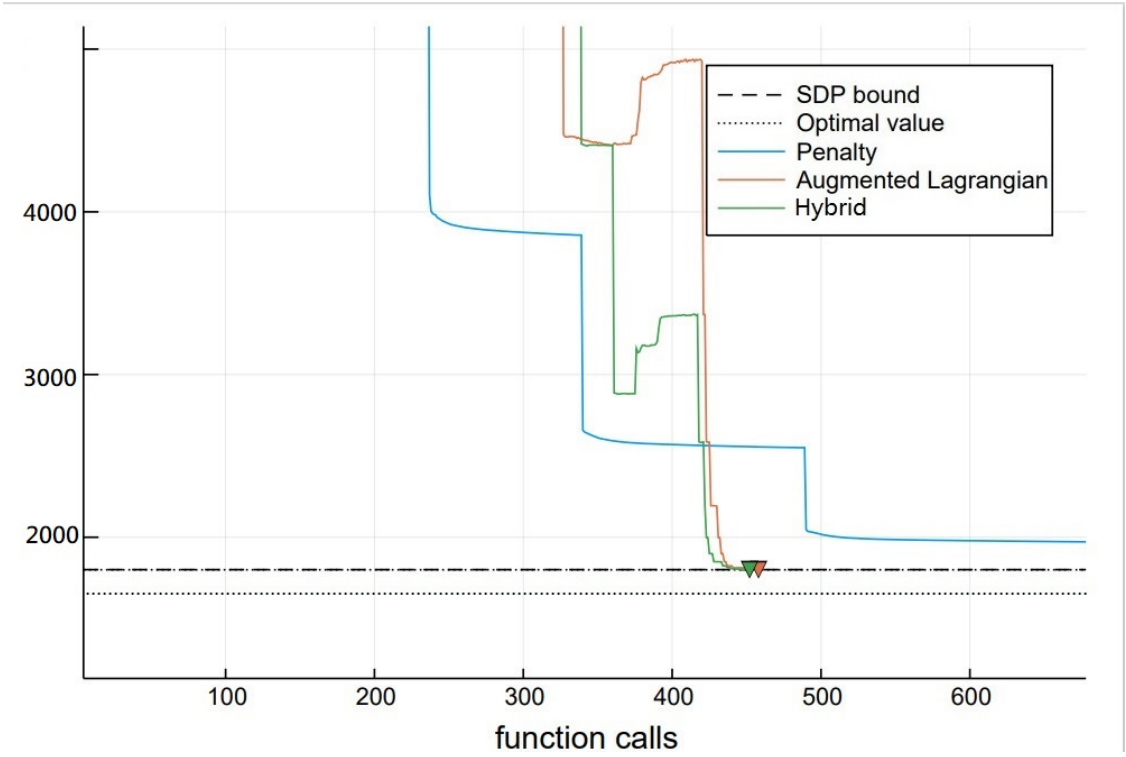

Figure 5.11: On graph (be120.8.2), bounds versus function calls for Augmented Lagrangian, penalty methods, and hybrid technique where y-axis represented the bound.

5.3.7 Numerical Results of Graphs (be150.3 , be150.8, be200.3 , be200.8 and be250.1)

In this section, a set of graphs taken from the Big Mac library where tested. To begin with, the graph (be150), as shown the Figure (5.12) where it shows the problem of (be150.3.2) and how to converge solutions according to the three methods mentioned in the Table (5.9). Also, the optimal value at 1781 while the goal bound at 1792. Where shown that the hybrid method had the first share of approaching the SDP bound at 1499 function calls followed by the Augmented lagrangian method at 1523 function calls and then the penalty method at 1570 function calls, which means that the hybrid method has saved us time and reached the SDP bound with the least number of function calls. As for the graph (be200.3) and (be200.8) as shown in Figure (5.13) where it shows the problem of (be200.3.1) where the same approach was applied in the previous problem.

The approach to the SDP bound was by the Augmented lagrangian method at 1443 function calls, followed by the hybrid method at 2043 function calls, and then the penalty method at 2480 function calls where the optimal value equal 2615 and the goal bound is 2633, see Table (5.10). Finally, for the graph (be250) as shown in Figure (5.14) where it shows the problem of (be250.5) , the same approach was applied in the previous problems. The approach to the SDP bound was by the hybrid method at 1661 function calls, followed by the Augmented lagrangian method at 2420 function calls, and then the penalty method at 2488 function calls where the optimal value equal 2409 and the goal bound is 2415, see Table (5.11).

Graphs	**Optimal value**	**Penalty**	**Augment-ed**	**Hybrid**	**The goal bound**
be150.3.1	18889	2000	1940	**1900**	18930.1
be150.3.2	17816	1980	**1360**	1743	17888.6
be150.3.3	17314	1238	1470	**1177**	17314.2
be150.3.4	19884	1660	**1420**	1502	19892.9
be150.3.5	17817	1570	1523	**1499**	17925
be150.3.6	16780	1802	2498	**1737**	16988.3
be150.3.7	18001	2679	**2039**	2044	18075.6
be150.3.8	18303	**1862**	2242	1982	18579.9
be150.3.9	12838	1859	2367	**1393**	13195.9
be150.3.10	17963	1359	**1207**	1313	18201.6
be150.8.1	27089	2469	2394	**2315**	27508.7
be150.8.2	29438	2270	2019	**1362**	27341.7
be150.8.3	26911	2970	1900	**1068**	29790.7
be150.8.4	28017	2450	2381	**2282**	27232.4
be150.8.5	29221	1922	2160	**1533**	28269.1
be150.8.6	31209	2802	1908	**1280**	29625.6
be150.8.7	29730	1988	**1639**	1849	31670.3
be150.8.8	25388	1799	**1695**	2236	30091
be150.8.9	28374	2782	**1480**	2687	25840.6
be150.8.10	26779	2589	**1844**	2450	28699.2
Total winner fcalls		**1**	**8**	**11**	

Table 5.9: The test types of graphs (be150.3 which has n=150 and d=0.3)and (be150.8 which has n=150 and d=0.8) was selected from the Biq Mac library by the function calls of (augmented Lagrangian, penalty, hybrid) methods.

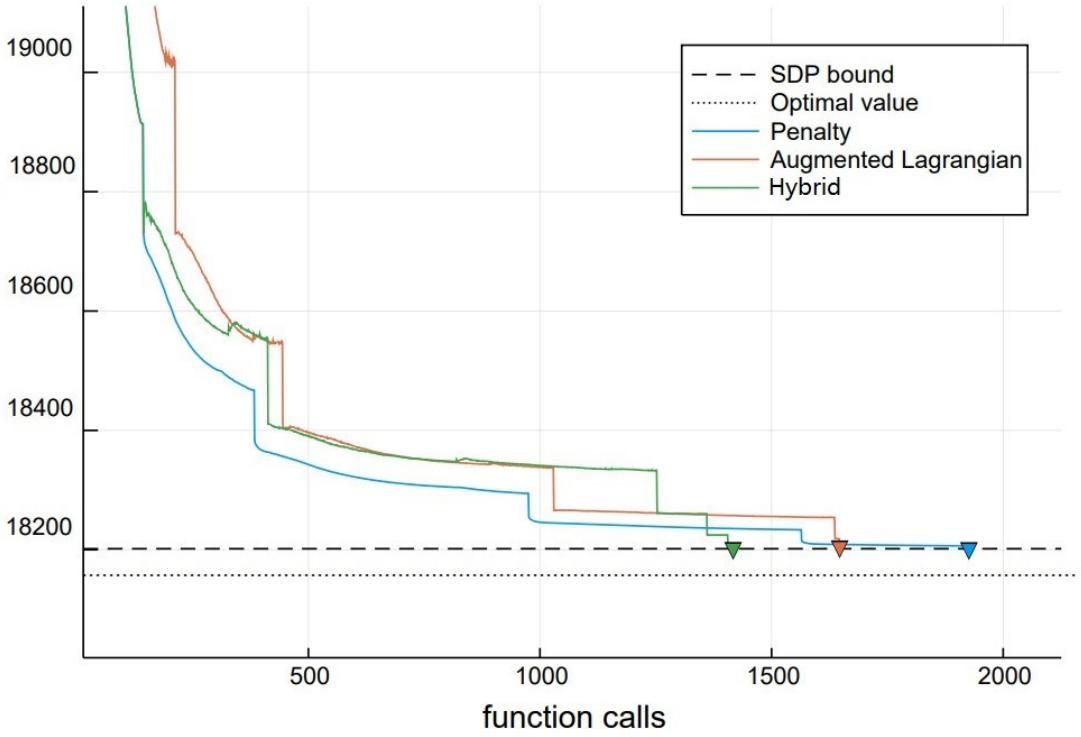

Figure 5.12: On graph (be150.3.5), bounds versus function calls for Augmented Lagrangian, penalty methods, and hybrid technique where y-axis represented the bound.

Graphs	**Optimal value**	**Penalty**	**Augment-ed**	**Hybrid**	**The goal bound**
be200.3.1	26153	2480	**1443**	2043	26338
be200.3.2	25027	**2730**	3043	2860	25513
be200.3.3	28031	1817	1850	**1490**	28284
be200.3.4	27434	**2047**	2280	2352	27836.8
be200.3.5	26355	1752	1700	**1694**	26749.7
be200.3.6	26146	2540	**2200**	2409	26571.3
be200.3.7	30483	1390	2294	**1382**	30535.8
be200.3.8	27355	1842	**1695**	1750	27816
be200.3.9	24683	2209	**1977**	2189	25168
be200.3.10	2642	2760	**2339**	2532	24458
be200.8.1	48534	2134	**2000**	2049	49086
be200.8.2	40821	2400	2890	**1976**	42060
be200.8.3	43207	1857	**1632**	1801	44322
be200.8.4	43757	1942	2201	**1490**	44504
be200.8.5	41482	1822	**1655**	1989	42404.7
be200.8.6	49492	1822	2955	**1757**	49683.7
be200.8.7	46828	**1725**	2200	2241	47541
be200.8.8	44502	1795	**1789**	1950	45617
be200.8.9	43241	1852	**1510**	1609	43664
be200.8.10	42832	1799	1782	**1793**	43596.9
Total winner fcalls		**3**	**10**	**7**	

Table 5.10: The test types of graphs (be200.3 which has n=200 and d=0.3)and (be200.8 which has n=200 and d=0.8) was selected from the Biq Mac library by the function calls of (augmented Lagrangian, penalty, hybrid) methods.

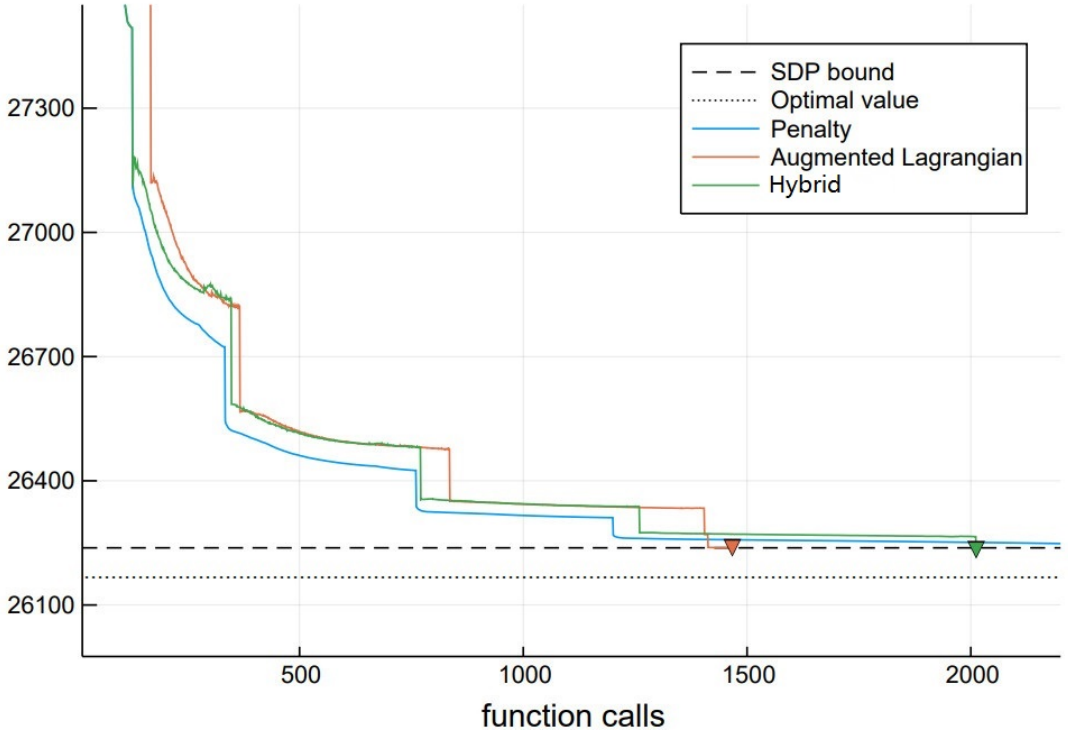

Figure 5.13: On graph be200.3.1, bounds versus function calls for Augmented Lagrangian, penalty methods, and hybrid technique where y-axis represented the bound.

$n = 250$, $density = 0.1$					
Graphs	**Optimal value**	**Penalty**	**Augment-ed**	**Hybrid**	**The goal bound**
be250.1	24076	2581	2652	**1846**	24098.8
be250.2	22540	3340	3406	**3242**	22574.6
be250.3	22923	3086	1741	**1690**	22923
be250.4	24649	2331	**2074**	2592	24656.9
be250.5	24091	2488	2420	**1661**	24159
be250.6	22735	1833	**1800**	2239	22797.9
be250.7	24095	1531	**1002**	1311	24095
be250.8	23801	2224	3630	**1759**	23858.4
be250.9	20051	2490	2688	**2479**	20139.9
be250.10	23159	**2286**	2784	2639	23222.6
Total winner fcalls		**1**	**3**	**6**	

Table 5.11: The test types of graphs be250 was selected from the Biq Mac library by the function calls of (augmented Lagrangian , penalty, hybrid) methods.

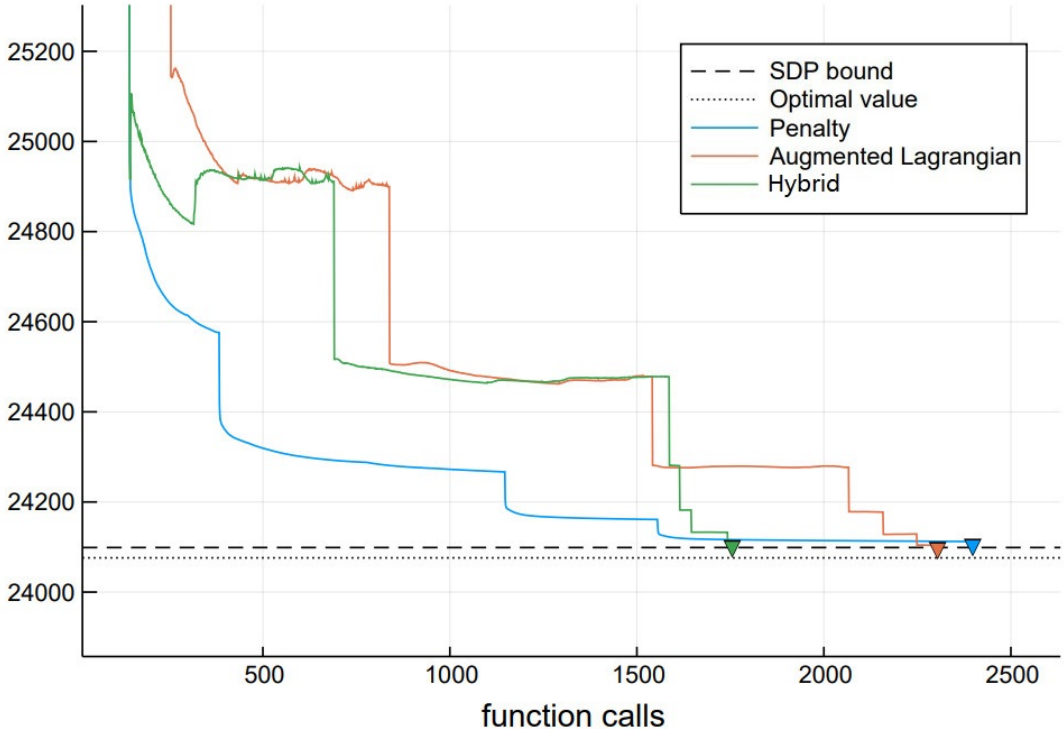

Figure 5.14: On graph (be250.5), bounds versus function calls for Augmented Lagrangian, penalty methods, and hybrid technique where y-axis represented the bound.

5.3.8 Numerical Results of Graphs (g05.60, g05.80 and g05.100)

The results of assessing different approaches on three types of a graphs which is (g05.60), (g05.80) and (g05.100). In fact, the graph of (g05.60) contains 60 nodes with 885 edges, the graph of (g05.80) contains 80 nodes with 1580 edges and the graph of (g05.100) contains 100 nodes with 2475 edges. Also, there are three models (g05.60.6), (g05.80.6) and (g05.100.4) are presented in Figures ((5.15), (5.16) and (5.17)) respectively. Besides, the tested was by three methods (penalty, Augmented Lagrangian, and a hybrid). We compared the results of the hybridization method, which combines the penalty method with the Augmented Lagrangian method where this new algorithm appears to converge better when the other two methods are employed separately. Also, three Tables ((5.12), (5.13) and (5.14)) as well as three figures (5.12),(5.13) and (5.14) respectively depicting the progress of the solution process, based on the three models

mentioned earlier. Furthermore, these figures above show that the SDP solution is equal to (531, 929, and 1431), whereas the semidefinite bound is equal to (533, 933, and 1441) respectively. According to the figure and table of model (g05.60.6), the hybrid method resulted was a faster convergence to the semidefinite bound at 166 function calls followed by the augmented Lagrangian method at 188 function calls, and then the penalty method at 500 function calls . Also, it was the same result for model (g05.80.6) where the hybrid method resulted was a faster convergence to the semidefinite bound at 151 function calls followed by the augmented Lagrangian method at 162 function calls, and then the penalty method at 467 function calls, while for model (g05.100.4) augmented Lagrangian method resulted was a faster convergence to the semidefinite bound at 243 function calls followed by the hybrid method at 256 function calls, and then the penalty method at 500 function calls. Finally, achieving the bound with the fewest number of function calls (saves time), which indicates that method is the fastest in convergence.

$n = 60$, $density = 0.5$					
Graphs	**Optimal value**	**Penalty**	**Augment-ed**	**Hybrid**	**The goal bound**
g05_60.0	536	699	545	**520**	537.3
g05_60.1	532	898	**467**	498	532.7
g05_60.2	529	432	298	**212**	532.7
g05_60.3	538	887	834	**576**	538.0
g05_60.4	527	450	335	**224**	530.8
g05_60.5	533	878	**334**	434	533.3
g05_60.6	531	500	188	**166**	533.6
g05_60.7	535	765	**390**	532	536.8
g05_60.8	530	439	298	**267**	532.7
g05_60.9	533	545	**276**	387	536.1
Total winner fcalls			**4**	**6**	

Table 5.12: The test types of graphs (g05.60) was selected from the Biq Mac library by the function calls of (augmented Lagrangian , penalty, hybrid) methods.

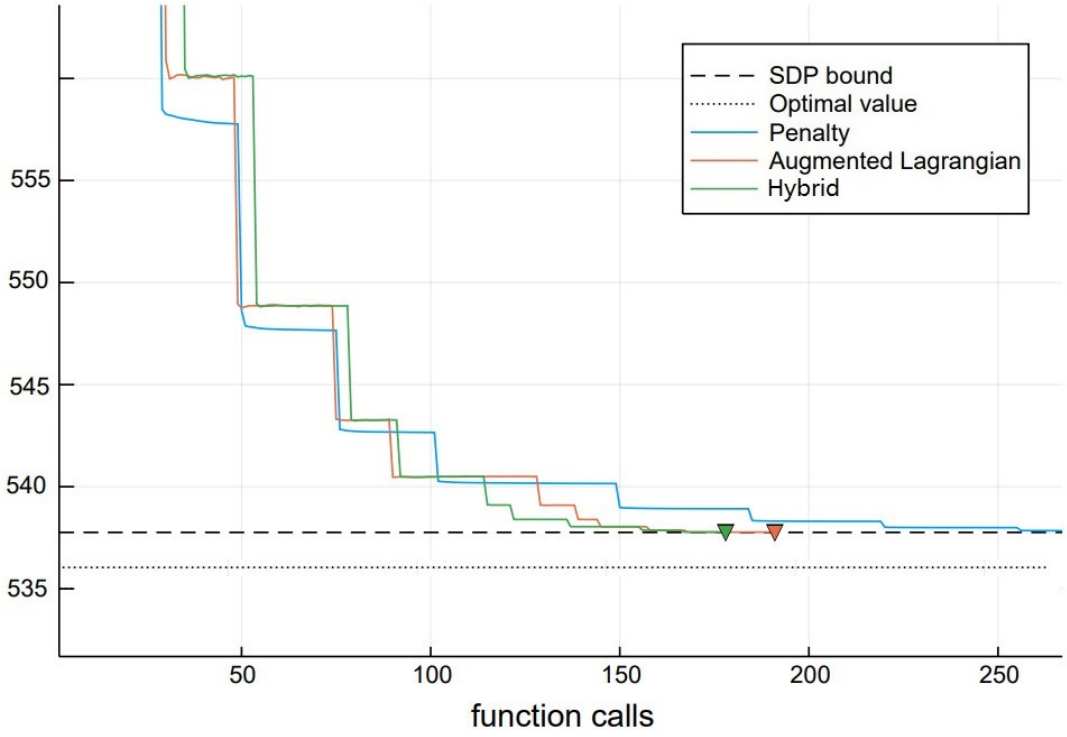

Figure 5.15: On graph (g05.60.6), bounds versus function calls for Augmented Lagrangian, penalty methods, and hybrid technique where y-axis represented the bound.

$n = 80$, $density = 0.5$					
Graphs	**Optimal value**	**Penalty**	**Augment-ed**	**Hybrid**	**The goal bound**
g05_80.0	929	655	**287**	388	934.4
g05_80.1	941	945	465	**378**	941.9
g05_80.2	934	746	**276**	356	937.4
g05_80.3	923	345	**145**	165	932.5
g05_80.4	932	645	245	**220**	936.7
g05_80.5	926	445	**185**	245	931.5
g05_80.6	929	467	162	**151**	933.4
g05_80.7	929	418	256	**245**	932.8
g05_80.8	925	476	256	**167**	930.7
g05_80.9	978	356	265	**145**	930.1
Total winner fcalls			**4**	**6**	

Table 5.13: The test types of graphs (g05.80) was selected from the Biq Mac library by the function calls of (augmented Lagrangian, penalty, hybrid) methods.

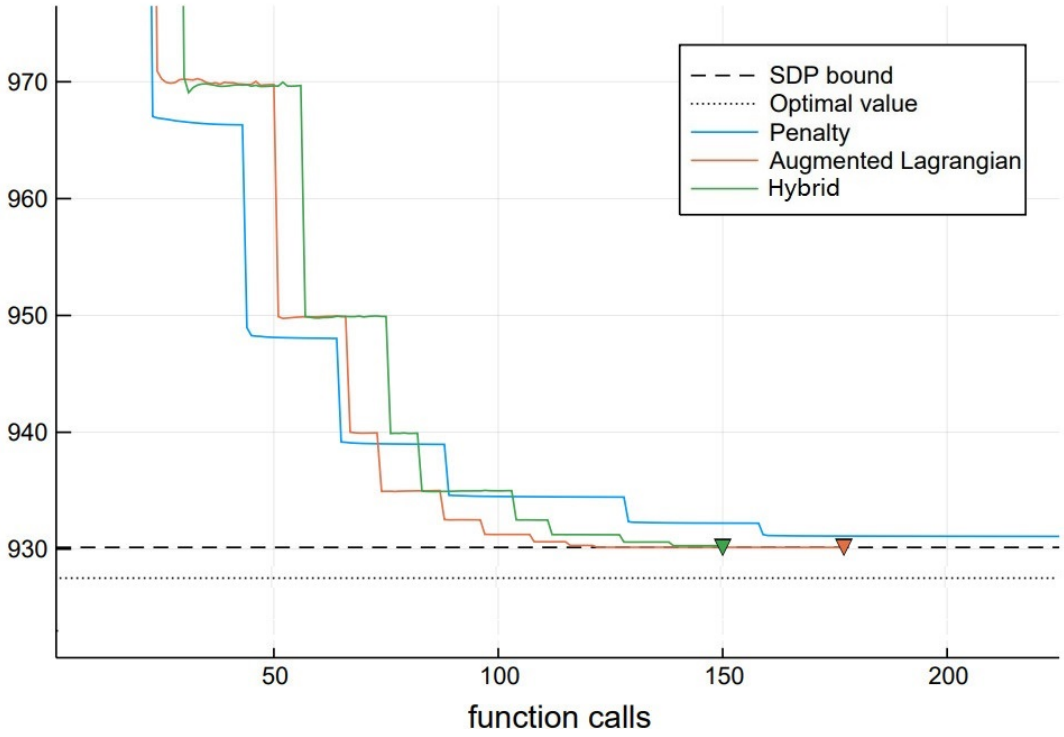

Figure 5.16: On graph (g05.80.6), bounds versus function calls for Augmented Lagrangian, penalty methods, and hybrid technique where y-axis represented the bound.

$n = 100$, $density = 0.5$					
Graphs	**Optimal value**	**Penalty**	**Augment-ed**	**Hybrid**	**The goal bound**
g05_100.0	1430	487	**187**	245	1442.3
g05_100.1	1425	245	200	**165**	1441.5
g05_100.2	1432	532	234	**232**	1440.1
g05_100.3	1424	287	187	**134**	1437.7
g05_100.4	1440	500	**234**	256	1446.0
g05_100.5	1436	534	243	**234**	1443.9
g05_100.6	1434	434	334	**243**	1442.7
g05_100.7	1431	534	266	**206**	1441.1
g05_100.8	1432	465	**214**	323	1442.4
g05_100.9	1430	245	265	**187**	1441.0
Total winner fcalls			**3**	**7**	

Table 5.14: The test types of graphs (g05.100) was selected from the Biq Mac library by the function calls of (augmented Lagrangian , penalty, hybrid) methods.

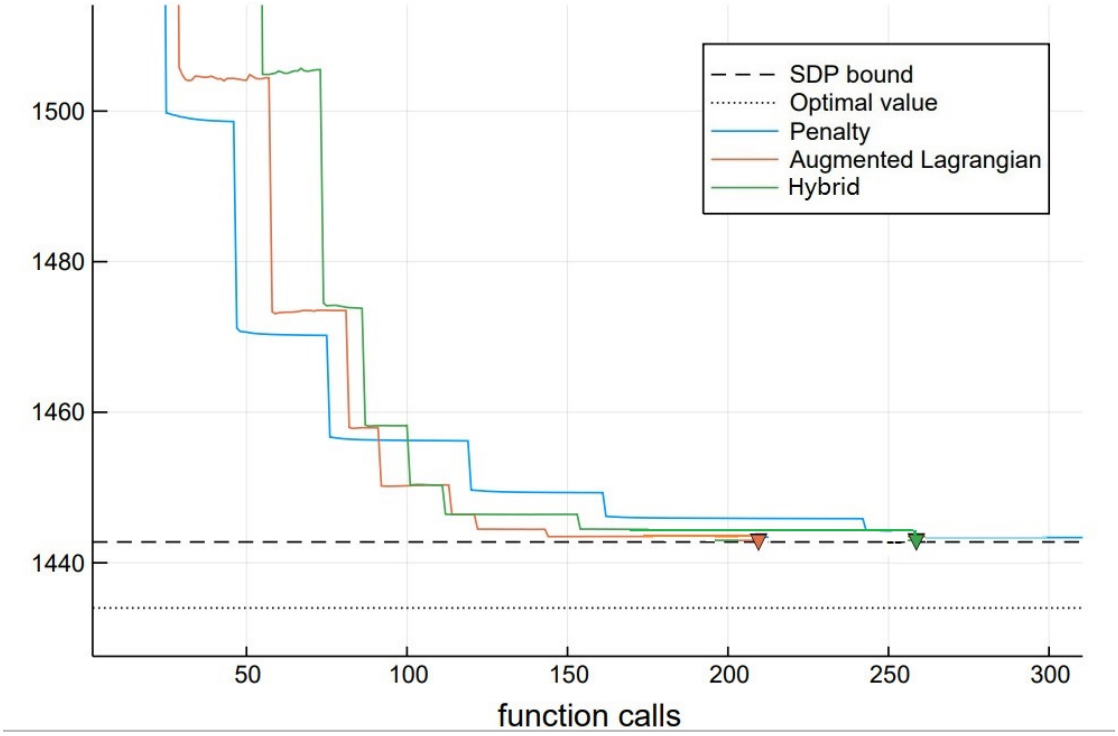

Figure 5.17: On graph (g05.100.4), bounds versus function calls for Augmented Lagrangian, penalty methods, and hybrid technique where y-axis represented the bound.

5.3.9 Numerical Results of Graphs (bqp.50, bqp.250 and bqp.500)

Following the previous approach in describing the results that obtained, and as shown in Tables ((5.15), and (5.16)), three other types of graphs were tested, namely (bqp.50), (bqp.250) and (bqp.500) . In fact, Figures ((5.18), (5.19) and (5.20)) refer to three selected samples of the results tables for the above graphs. Let's start with the first sample, which is (bqp.50.1), where the hybrid method was faster in convergence with the semidefinite bound at 4440 function calls, followed by the augmented Lagrangian method at the 5101 function calls, and then the penalty method in the 5220 function calls, knowing that the optimal value for this sample was 2089 and the SDP bound was 2089. The same result was for the sample (bqp.250.7) where the hybrid method was faster in convergence with the semidefinite bound at 2651 function, followed by the Augmented Lagrangian

method and Penalty method at the 3000 function calls, knowing that the optimal value for this type was 4675 and the SDP bound was 4678. Finally, for the sample (bqp.500.1), the result was similar to the previous samples in terms of the convergence and saving time of the hybrid method to the the SDP bound at 2172 function calls followed by the augmented Lagrangian method at the 2397 function calls, and then the penalty method in the 3501 function calls, knowing that the optimal value for this sample was 1156 and the SDP bound was 1210. Faster convergence and saving time are our goal to reach the optimal solution, which is found at the end the general summary of the concept of optimization.

Graphs	**Optimal value**	**Penalty**	**Augment-ed**	**Hybrid**	**The goal bound**
bqp50_1	2098	5220	5101	**4440**	2098
bqp50_2	3702	765	**498**	665	3702.5
bqp50_3	4626	498	487	**276**	4626.8
bqp50_4	3544	676	598	**465**	3544.4
bqp50_5	4012	334	656	**298**	4012.9
bqp50_6	3693	487	455	**287**	3693.9
bqp50_7	4520	**345**	798	445	4520.3
bqp50_8	4216	699	645	**287**	4216
bqp50_9	3780	**487**	556	498	3780
bqp50_10	3507	755	**498**	556	3507
bqp250_1	45607	3002	**2765**	2891	45622.5
bqp250_2	44810	2789	3213	**2001**	44866.9
bqp250_3	49037	2415	**2075**	2091	49037.8
bqp250_4	41274	2984	4267	**2780**	41341.9
bqp250_5	47961	4856	3123	**2651**	47961.3
bqp250_6	41014	4961	4764	**4470**	41356.3
bqp250_7	46757	3000	3000	**2651**	46783.9
bqp250_8	35726	3700	3499	**3243**	36537
bqp250_9	49003	48916	2733	**2025**	2292
bqp250_10	40442	5127	5187	**4783**	40614
Total winner fcalls		**2**	**4**	**14**	

Table 5.15: The test types of graphs (bqp50 which has n=50 ,d=0.1) and (bqp250 which has n=250, d=0.1) was selected from the Biq Mac library by the function calls of (augmented Lagrangian, penalty, hybrid) methods.

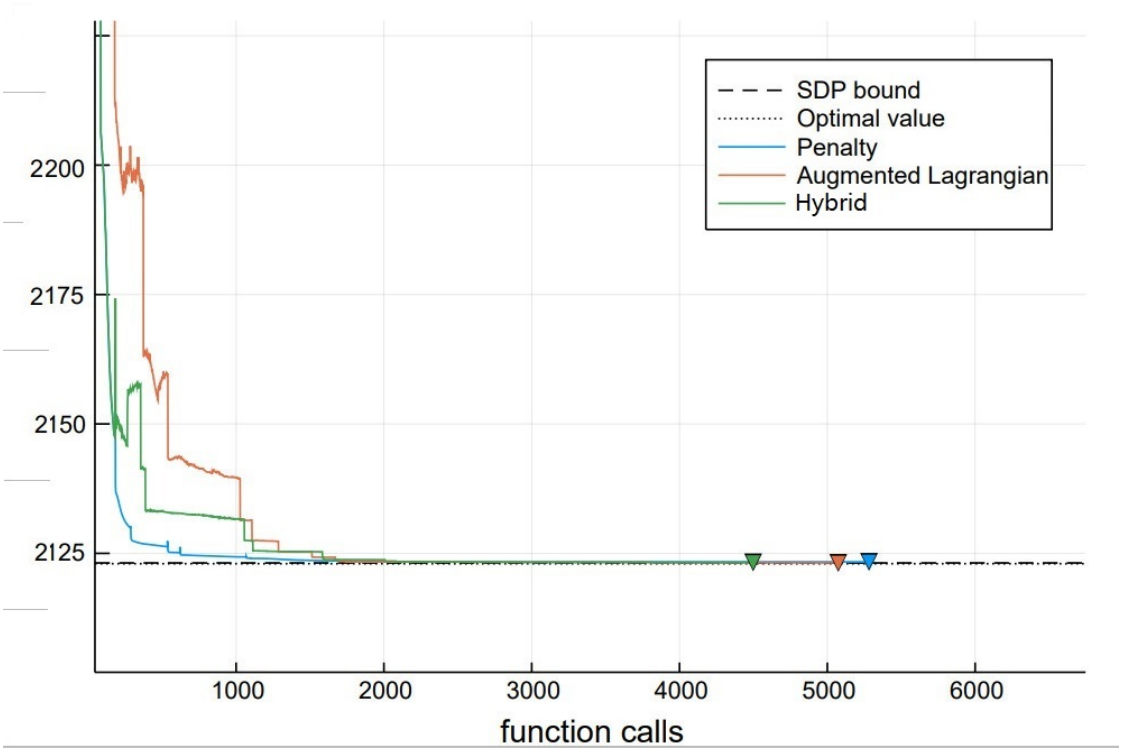

Figure 5.18: On graph (bqp50.1), bounds versus function calls for augmented Lagrangian, penalty approaches, and hybrid technique.

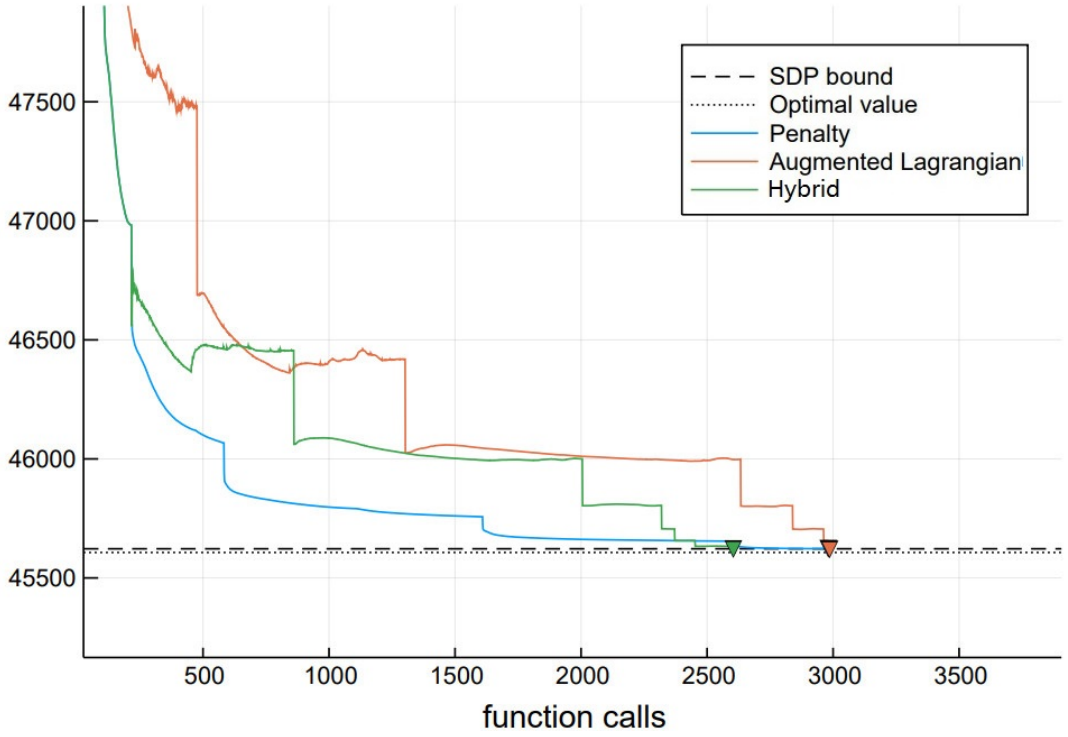

Figure 5.19: On graph (bqp 250.7), bounds versus function calls for Augmented Lagrangian, penalty methods, and hybrid technique where y-axis represented the bound.

$n = 500$, $density = 0.1$					
Graphs	**Optimal value**	**Penalty**	**Augment-ed**	**Hybrid**	**The goal bound**
bqp500_1	116586	3501	2397	**2172**	121001.5
bqp500_2	128223	3862	3145	**2703**	131282.8
bqp500_3	130812	4143	4156	**3875**	133329
bqp500_4	130097	3984	**3674**	3871	133872
bqp500_5	125487	3374	**2971**	3984	128662.5
bqp500_6	121772	4653	4873	**3895**	125654
bqp500_7	122201	3651	**2000**	3981	126338
bqp500_8	123559	4976	**3900**	3990	128012.7
bqp500_9	120798	4576	3872	**2873**	125368.7
bqp500_10	130619	3345	2675	**2482**	133360.7
Total winner fcalls		**0**	**4**	**6**	

Table 5.16: The test types of graphs (bqp500) was selected from the Biq Mac library by the function calls of (augmented Lagrangian , penalty, hybrid) methods.

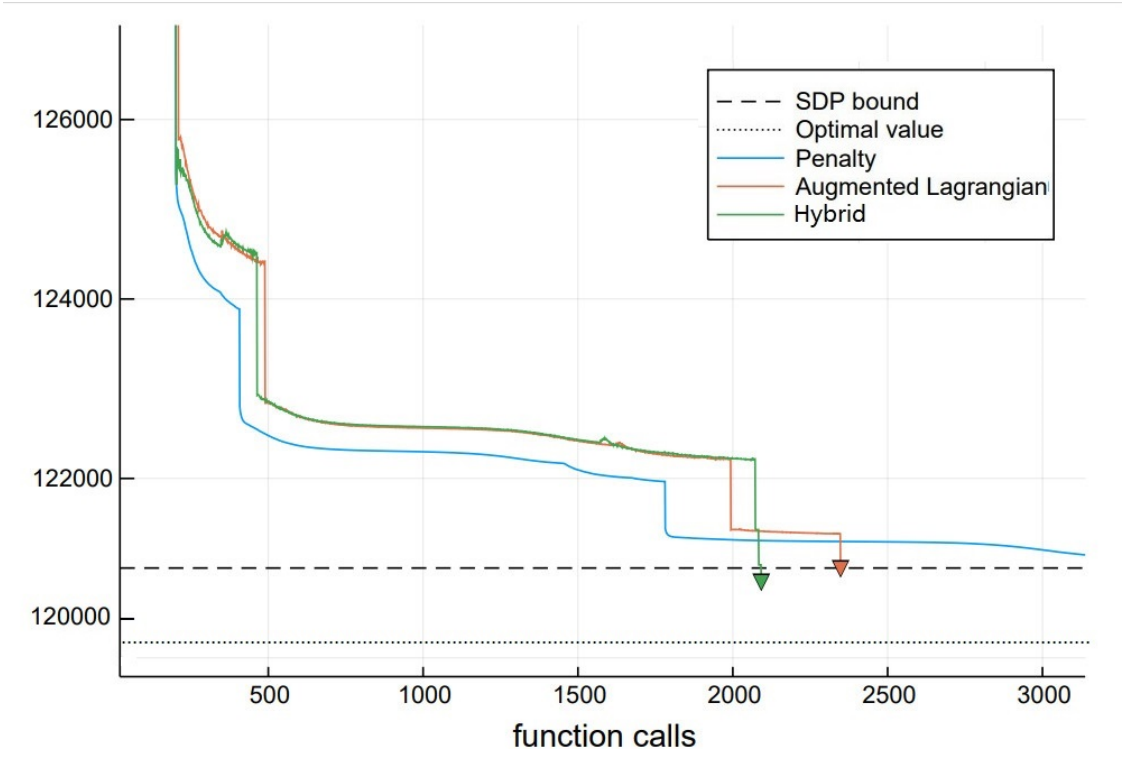

Figure 5.20: On graph(bqp500.1), bounds versus function calls for Augmented Lagrangian, penalty methods, and hybrid technique where y-axis represented the bound.

5.3.10 Numerical Results of Graphs (bqp.100)

As in our previous tests, we will review the problem (bqp.100) which was selected from the Biq Mac library. As shown in the Figure (5.21) of the sample (bqp.100.10) the exact solution K-CLUSTER in this figure appears to be equal to 12565, while the semidefinite bound is equal to 12566.9 . When comparing the results of the three methods (penalty method, augmented Lagrangian method, and a hybrid method), we noticed that the chances of winning by approaching to semidefinite bound was for penalty method, where it reached the bound by 491 function calls , while the hybrid method had less luck than that, as it reached 540 function calls and then followed by augmented Lagrangian method by 663 function calls . The results of all the results for the problem (bqp.100) are mentioned in Table (5.17) with meaning that getting to the bound with the fewest number of function calls is saving time, and this is our goal that we need to achieve.

$n = 100$, $density = 0.1$					
Graphs	**Optimal value**	**Penalty**	**Augment-ed**	**Hybrid**	**The goal bound**
bqp100_1	7970	**471**	623	537	7970.3
bqp100_2	11036	**322**	1010	1003	11036.4
bqp100_3	12723	**441**	794	535	12723.4
bqp100_4	10368	453	**418**	487	10368.6
bqp100_5	9083	440	**379**	401	9084.3
bqp100_6	10210	**254**	1433	453	10211.8
bqp100_7	10125	619	615	**333**	10125
bqp100_8	11435	503	499	**311**	11436
bqp100_9	11455	**419**	771	514	11455.4
bqp100_10	12565	**491**	663	540	12566.9
Total winner fcalls		**6**	**2**	**2**	

Table 5.17: The test types of graphs (bqp100) was selected from the Biq Mac library by the function calls of (augmented Lagrangian, penalty, hybrid) methods .

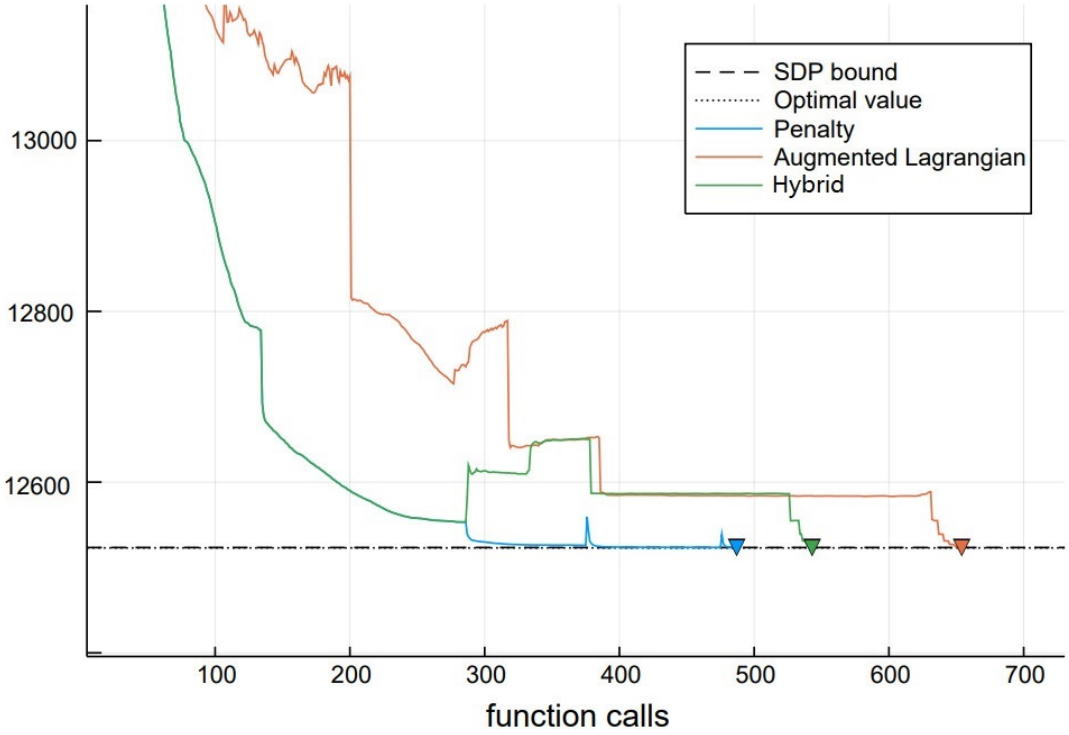

Figure 5.21: On graph (bqp100.10), bounds versus function calls for Augmented Lagrangian, penalty methods, and hybrid technique where y-axis represented the bound.

5.4 Conclusions

The dissertation's goals and objectives were met, and the following information was gleaned:

1. Development a new relaxation of the feasible region to provide the Augmented lagrangian method based on penalty method.

2. The most important applications used in the real world and the extent to which these applications overlap with other features have been shown in line with the available updates.

3. The new approach was evaluated using a variety of graphs from the Biq Mac library. These graphs included a variety of characteristics, including a huge number of nodes and edges.

4. The results indicated that the Augmented lagrangian method was preferable than the Penalty method in terms of achieving the goal bound .

5. We also put the Hybrid technique to the test, which alternates between the penalty and inequality Augmented lagrangian method. The Hybrid technique performed better than the two strategies independently, according to the findings.

6. It has been proven that the K-CLUSTER combinatorial optimization problems is NP-Hardness problem in graph clustering with large scale of variables.

7. A new algorithm has been developed by which the bound is improved and it works on NP-Hardness problem.

8. The most important applications of clustering in real-life have been presented.

9. Develop theoretical convergence properties.

5.5 Future Work

According to the concept of our work, future plans can be based on the following developments:

1. According to the results and conclusions, a hybrid algorithm that combines the penalty method with the augmented Lagrangian method to minimize convergence time may be developed. This would be feasible by offering a new technique to switch between the two methods during certain moments of the bounding operation.

2. Other NP-hard problems, such as general binary quadratic problems, can be used in future work.

3. More graphs may be tried, and the results can be extracted to have a better idea of how well the method works.

4. It is possible to investigate large-scale binary quadratic problems.

5. Theoretical convergence properties can help us better understand the bounding method.

6. A new relaxation with Penalty methods to get the Augmented Lagrange method by different parameter.

7. This work can be used in the formation new solvers as CSDP Borchers (1999), DSDP Benson and Ye (2005) , PENLAB Fiala, Koˇcvara, and Stingl (2005) , SDPA Nakata (2010) ,and BiqMac Wiegele (2007) [1].

REFERENCES

[1] Ahmed Al-Jilawi. *Solving the Semidefinite Programming Relaxation of Max-cut Using an Augmented Lagrangian Method*. Northern Illinois University, 2019.

[2] Leonardo Jesus Almeida and Alneu de Andrade Lopes. An ultra-fast modularity-based graph clustering algorithm. In *Proceedings 14th Portuguese Conference on Artificial Intelligence (EPIA)-Web and Network Intelligence Track*, pages 1–9, 2009.

[3] Ahmed Alridha and Ahmed Sabah Al-Jilawi. Mathematical programming computational for solving np-hardness problem. In *Journal of Physics: Conference Series*, volume 1818, page 012137. IOP Publishing, 2021.

[4] Ahmed Alridha, Abbas Musleh Salman, and Ahmed Sabah Al-Jilawi. The applications of np-hardness optimizations problem. In *Journal of Physics: Conference Series*, volume 1818, page 012179. IOP Publishing, 2021.

[5] Niclas Andréasson, Michael Patriksson, and Anton Evgrafov. *An introduction to continuous optimization: foundations and fundamental algorithms*. Courier Dover Publications, 2020.

[6] Neculai Andrei. Penalty and augmented lagrangian methods. In *Continuous Nonlinear Optimization for Engineering Applications in GAMS Technology*, pages 185–201. Springer, 2017.

[7] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media, 2012.

[8] Hideyuki Azegami. Shape optimization problems of domain variation type. In *Shape Optimization Problems*, pages 427–566. Springer, 2020.

[9] Elham Photoohi Bafghi. Clustering of customers based on shopping behavior and employing genetic algorithms. *Engineering, Technology & Applied Science Research*, 7(1):1420–1424, 2017.

[10] Rangaswami Balakrishnan and Kanna Ranganathan. *A textbook of graph theory*. Springer Science & Business Media, 2012.

[11] Venkataramanan K Balakrishnan. *Introductory discrete mathematics*. Courier Corporation, 2012.

[12] Dimitri Bertsekas. *Convex optimization algorithms*. Athena Scientific, 2015.

[13] Dimitri P Bertsekas, W Hager, and O Mangasarian. Nonlinear programming. athena scientific belmont. *Massachusets, USA*, 1999.

[14] Dario Andrea Bini, Fabio Di Benedetto, Eugene Tyrtyshnikov, and Marc Van Barel. *Structured Matrices in Numerical Linear Algebra: Analysis, Algorithms and Applications*, volume 30. Springer, 2019.

[15] Ernesto G Birgin and José Mario Martínez. *Practical augmented Lagrangian methods for constrained optimization*. SIAM, 2014.

[16] Michael Birsak. *Discrete optimization on graphs and grids for the creation of navigational and artistic imagery*. PhD thesis, Wien, 2018.

[17] Grigoriy Blekherman, Pablo A Parrilo, and Rekha R Thomas. *Semidefinite optimization and convex algebraic geometry*. SIAM, 2012.

[18] János Bognár. *Indefinite inner product spaces*, volume 78. Springer Science & Business Media, 2012.

[19] Béla Bollobás. *Graph theory: an introductory course*, volume 63. Springer Science & Business Media, 2012.

[20] Béla Bollobás. *Modern graph theory*, volume 184. Springer Science & Business Media, 2013.

[21] Brian Borchers. Csdp, ac library for semidefinite programming. *Optimization methods and Software*, 11(1-4):613–623, 1999.

[22] Jan Brinkhuis. *Convex analysis for optimization*. Springer, 2020.

[23] Fred Buckley and Marty Lewinter. *Introductory Graph Theory with Applications*. Waveland Press, 2013.

[24] Sergiy Butenko and Panos M Pardalos. *Numerical methods and optimization: An introduction*. CRC Press, 2014.

[25] Ilya Bychkov, Valery A Kalyagin, Panos M Pardalos, and Oleg Prokopyev. *Network Algorithms, Data Mining, and Applications: NET, Moscow, Russia, May 2018*, volume 315. Springer Nature, 2020.

[26] Moses Charikar. On semidefinite programming relaxations for graph coloring and vertex cover. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 616–620, 2002.

[27] Diego Cifuentes, Corey Harris, and Bernd Sturmfels. The geometry of sdp-exactness in quadratic optimization. *Mathematical programming*, 182(1):399–428, 2020.

[28] William Cook. Computing in combinatorial optimization. In *Computing and Software Science*, pages 27–47. Springer, 2019.

[29] Derek G Corneil and Yehoshua Perl. Clustering and domination in perfect graphs. *Discrete Applied Mathematics*, 9(1):27–39, 1984.

[30] Richard W Cottle and Gerd Infanger. Harry markowitz and the early history of quadratic programming. In *Handbook of Portfolio Construction*, pages 179–211. Springer, 2010.

[31] Etienne De Klerk. *Aspects of semidefinite programming: interior point algorithms and selected applications*, volume 65. Springer Science & Business Media, 2006.

[32] Stephan Dempe and Alain Zemkoho. *Bilevel optimization*. Springer, 2020.

[33] Sushil C Dimri, Preeti Malik, and Mangey Ram. Algorithms. In *Algorithms*. De Gruyter, 2021.

[34] Urmila M Diwekar. *Introduction to applied optimization*, volume 22. Springer Nature, 2020.

[35] Moshe Dror. *Arc routing: theory, solutions and applications*. Springer Science & Business Media, 2012.

[36] Dingzhu Du, Panos M Pardalos, and Zhao Zhang. *Nonlinear Combinatorial Optimization*, volume 147. Springer, 2019.

[37] Mahmoud Mahmoud El-Alem. *A global convergence theory for a class of trust region algorithms for constrained optimization*. PhD thesis, Rice University, 1988.

[38] Alain Faye and Frédéric Roupin. Partial lagrangian relaxation for general quadratic programming. *4OR*, 5(1):75–88, 2007.

[39] AV Fiacco and GP McCormick. Sequential unconstrained minimization techniques for nonlinear programming. *A primal dual method. Munagement Sci*, 10, 1968.

[40] Virginie Gabrel and Céline Murat. Robustness and duality in linear programming. *Journal of the Operational Research Society*, 61(8):1288–1296, 2010.

[41] Greg Gamble and Jamie Simpson. Symmetric difference-free and symmetric difference-closed collections of sets. *Graphs and Combinatorics*, 31(1):127–130, 2015.

[42] Islem Gammoudi, Mohamed Ali Mahjoub, and Fethi Guerdelli. Unsupervised image segmentation based graph clustering methods. *Computación y Sistemas*, 24(3), 2020.

[43] Guojun Gan, Chaoqun Ma, and Jianhong Wu. *Data clustering: theory, algorithms, and applications*. SIAM, 2020.

[44] Michel X Goemans. Semidefinite programming in combinatorial optimization. *Mathematical Programming*, 79(1):143–161, 1997.

[45] Andrés Gómez. Strong formulations for conic quadratic optimization with indicator variables. *Mathematical Programming*, 188(1):193–226, 2021.

[46] Nicholas Gould. An introduction to algorithms for continuous optimization, 2006.

[47] Jan Friso Groote and Kim Guldstrand Larsen. Tools and algorithms for the construction and analysis of systems: 27th international conference, tacas 2021, held as part of the european joint conferences on theory and practice of software, etaps 2021, luxembourg city, luxembourg, march 27–april 1, 2021, proceedings, part i, 2021.

[48] Jonathan L Gross, Jay Yellen, and Mark Anderson. *Graph theory and its applications*. Chapman and Hall/CRC, 2018.

[49] PC Haarhoff and JD Buys. A new method for the optimization of a nonlinear function subject to nonlinear constraints. *The Computer Journal*, 13(2):178–184, 1970.

[50] MR Hestenes. Multipliers and gradient methods—in computing met h ods in optimization problems, vol. 2, la zadeh, lw neustadt, av balakrishnan eds, 1969.

[51] Vladimír Holỳ, Ondřej Sokol, and Michal Černỳ. Clustering retail products based on customer behaviour. *Applied Soft Computing*, 60:752–762, 2017.

[52] Alston S Householder. *The theory of matrices in numerical analysis*. Courier Corporation, 2013.

[53] Abdelazim G Hussien, Aboul Ella Hassanien, Essam H Houssein, Mohamed Amin, and Ahmad Taher Azar. New binary whale optimization algorithm for discrete optimization problems. *Engineering Optimization*, 52(6):945–959, 2020.

[54] José Pedro Iglesias, Carl Olsson, and Fredrik Kahl. Global optimality for point set registration using semidefinite programming. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8287–8295, 2020.

[55] Kevin Igwe, Nelishia Pillay, and Christopher Rae. Solving the 8-puzzle problem using genetic programming. In *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, pages 64–67, 2013.

[56] Ali Jazayeri and Hiroki Sayama. A polynomial-time deterministic approach to the traveling salesperson problem. *arXiv preprint arXiv:1608.01716*, 2016.

[57] Qiang Ji and Ying Fan. Evolution of the world crude oil market integration: A graph theory analysis. *Energy Economics*, 53:90–100, 2016.

[58] Nathaniel Johnston. Advanced linear and matrix algebra.

[59] Nathaniel Johnston et al. Introduction to linear and matrix algebra. 2021.

[60] E Kay. Introductory graph theory. *Journal of the Operational Research Society*, 29(12):1245–1245, 1978.

[61] Isaac Klapper, Daniel B Szyld, and Kai Zhao. *Metabolic Networks, Elementary Flux Modes, and Polyhedral Cones*. SIAM, 2021.

[62] Ulrich Knauer and Kolja Knauer. *Algebraic graph theory*. de Gruyter, 2019.

[63] Mykel J Kochenderfer and Tim A Wheeler. *Algorithms for optimization*. Mit Press, 2019.

[64] Marek Kowalski and Jacek Naruniec. Face alignment using k-cluster regression forests with weighted splitting. *IEEE Signal Processing Letters*, 23(11):1567–1571, 2016.

[65] Nathan Krislock, Jérôme Malick, and Frédéric Roupin. Improved semidefinite branch-and-bound algorithm for k-cluster. *Available online as preprint hal-00717212*, 2012.

[66] Nathan Krislock, Jérôme Malick, and Frédéric Roupin. Computational results of a semidefinite branch-and-bound algorithm for k-cluster. *Computers & Operations Research*, 66:153–159, 2016.

[67] Sanjay Kumar. Monitoring novel corona virus (covid-19) infections in india by cluster analysis. *Annals of Data Science*, 7:417–425, 2020.

[68] Guanghui Lan and Renato DC Monteiro. Iteration-complexity of first-order augmented lagrangian methods for convex programming. *Mathematical Programming*, 155(1-2):511–547, 2016.

[69] Marko Lange. *Semidefinite relaxation approaches for the quadratic assignment problem*. PhD thesis, Technische Universität Hamburg, 2016.

[70] Claude Lemaréchal and François Oustry. *Semidefinite relaxations and Lagrangian duality with application to combinatorial optimization*. PhD thesis, INRIA, 1999.

[71] Wenjun Li, Yang Ding, Yongjie Yang, R Simon Sherratt, Jong Hyuk Park, and Jin Wang. Parameterized algorithms of fundamental np-hard problems: a survey. *Human-Centric Computing and Information Sciences*, 10(1):1–24, 2020.

[72] Zhouchen Lin, Huan Li, and Cong Fang. *Accelerated optimization for machine learning*. Springer, 2020.

[73] Junjun Liu and Wenzheng Li. Greedy permuting method for genetic algorithm on traveling salesman problem. In *2018 8th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pages 47–51. IEEE, 2018.

[74] Bas Lodewijks. Mapping np-hard and np-complete optimisation problems to quadratic unconstrained binary optimisation problems. *arXiv preprint arXiv:1911.08043*, 2019.

[75] Andrea Lodi and Viswanath Nagarajan. *Integer Programming and Combinatorial Optimization: 20th International Conference, IPCO 2019, Ann Arbor, MI, USA, May 22-24, 2019, Proceedings*, volume 11480. Springer, 2019.

[76] Cheng Lu, Shu-Cherng Fang, Qingwei Jin, Zhenbo Wang, and Wenxun Xing. Kkt solution and conic relaxation for solving quadratically constrained quadratic programming problems. *SIAM Journal on Optimization*, 21(4):1475–1490, 2011.

[77] David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*, volume 2. Springer, 1984.

[78] Ramtin Madani, Mohsen Kheirandishfard, Javad Lavaei, and Alper Atamtürk. Penalized semidefinite programming for quadratically-constrained quadratic optimization. *Journal of Global Optimization*, 78(3):423–451, 2020.

[79] Ananta K Majee and Andreas Prohl. Optimal strong rates of convergence for a space-time discretization of the stochastic allen–cahn equation with multiplicative noise. *Computational Methods in Applied Mathematics*, 18(2):297–311, 2018.

[80] Jérôme Malick and Frédéric Roupin. Solving k-cluster problems to optimality with semidefinite programming.

[81] Jérôme Malick and Frédéric Roupin. Numerical study of semidefinite bounds for the k-cluster problem. *Electronic Notes in Discrete Mathematics*, 36:399–406, 2010.

[82] Jérôme Malick and Frédéric Roupin. *Solving k-cluster problems to optimality using adjustable semidefinite programming bounds*. PhD thesis, CEDRIC Lab/CNAM, 2010.

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

[84] Jiali Mao, Qiuge Song, Cheqing Jin, Zhigang Zhang, and Aoying Zhou. Online clustering of streaming trajectories. *Frontiers of Computer Science*, 12(2):245–263, 2018.

[85] B Martin-Barragan, RE Lillo, and J Romo. Functional boxplots based on epigraphs and hypographs. *Journal of Applied Statistics*, 43(6):1088–1103, 2016.

[86] Paul D McNicholas and Peter A Tait. *Data science with Julia*. Chapman and Hall/CRC, 2019.

[87] Shashi Kant Mishra and Bhagwat Ram. *Introduction to linear programming with MATLAB*. CRC Press, 2017.

[88] Shashi Kant Mishra and Bhagwat Ram. *Introduction to Unconstrained Optimization with R*. Springer Nature, 2019.

[89] Jason J Molitierno. *Applications of combinatorial matrix theory to Laplacian matrices of graphs*. CRC Press, 2016.

[90] Anna Navrotskaya and Victor Il'ev. A local search for a graph clustering problem. In *AIP Conference Proceedings*, volume 1776, page 050004. AIP Publishing LLC, 2016.

[91] Sukanta Nayak. *Fundamentals of Optimization Techniques with Algorithms*. Academic Press, 2020.

[92] Pekka Neittaanmäki, Sergey Repin, and Tero Tuovinen. *Mathematical modeling and optimization of complex structures*. Springer, 2016.

[93] Mark Ed Newman, Albert-László Ed Barabási, and Duncan J Watts. *The structure and dynamics of networks*. Princeton university press, 2006.

[94] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[95] Panos M Pardalos, Mikhail Yu Khachay, and Alexander Kazakov. *Mathematical Optimization Theory and Operations Research: 20th International Conference, MOTOR 2021, Irkutsk, Russia, July 5-10, 2021: Proceedings*, volume 12755. Springer Nature, 2021.

[96] Alberto Passuello. *Semidefinite programming in combinatorial optimization with applications to coding theory and geometry*. PhD thesis, Université Sciences et Technologies-Bordeaux I, 2013.

[97] Michael JD Powell. A method for nonlinear constraints in minimization problems. *Optimization*, pages 283–298, 1969.

[98] Nagabhushana Prabhu and W He. A quartic nonlinear optimization method. *Neural, Parallel & Scientific Computations*, 10(2):209–226, 2002.

[99] Md Saidur Rahman et al. *Basic graph theory*. Springer, 2017.

[100] K Sobha Rani, KVSVN Raju, and V Valli Kumari. Application of clustering to analyze academic social networks. *International Journal of Web & Semantic Technology*, 4(2):9, 2013.

[101] Xiaoying Ren, Yingmin Wang, Lichen Zhang, and Qi Wang. Robust beamforming based on weighted vector norm regularization. *IEEE Access*, 9:88894–88901, 2021.

[102] R Tyrell Rockafellar. The multiplier method of hestenes and powell applied to convex programming. *Journal of Optimization Theory and applications*, 12(6):555–562, 1973.

[103] Ralph Tyrell Rockafellar. *Convex analysis*. Princeton university press, 2015.

[104] Ryan A Rossi, David F Gleich, Assefaw H Gebremedhin, and Md Mostofa Ali Patwary. Fast maximum clique algorithms for large graphs. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 365–366, 2014.

[105] Tim Roughgarden. *Algorithms Illuminated*. Soundlikeyourself publishing, 2017.

[106] Tim Roughgarden. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2021.

[107] Alexander Rybalov. On generic np-completeness of the graph clustering problem. In *Journal of Physics: Conference Series*, volume 1441, page 012167. IOP Publishing, 2020.

[108] Guillaume Sagnol. On the semidefinite representation of real functions applied to symmetric matrices. *Linear Algebra and its Applications*, 439(10):2829–2843, 2013.

[109] Tuhin Sahai. Dynamical systems theory and algorithms for np-hard problems. In *Proceedings of the Workshop on Dynamics, Optimization and Computation held in honor of the 60th birthday of Michael Dellnitz*, pages 183–206. Springer, 2020.

[110] Rudolf Scitovski, Kristian Sabo, Francisco Martínez-Álvarez, and Šime Ungar. Cluster analysis and applications, 2021.

[111] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[112] Kuldeep Singh, Harish Kumar Shakya, and Bhaskar Biswas. Clustering of people in social network based on textual similarity. *Perspectives in Science*, 8:570–573, 2016.

[113] Steven S Skiena. Graph problems: Np-hard. In *The Algorithm Design Manual*, pages 585–620. Springer, 2020.

[114] Defeng Sun, Kim-Chuan Toh, Yancheng Yuan, and Xin-Yuan Zhao. Sdpnal+: A matlab software for semidefinite programming with bound constraints (version 1.0). *Optimization Methods and Software*, 35(1):87–115, 2020.

[115] Bruce Torrence et al. Seeing the symmetric difference. In *Bridges 2020 Conference Proceedings*, pages 387–390. Tessellations Publishing, 2020.

[116] Levent Tunçel. *Polyhedral and semidefinite programming methods in combinatorial optimization*, volume 27. American Mathematical Soc., 2016.

[117] Robert J Vanderbei. *Linear programming: foundations and extensions*, volume 285. Springer Nature, 2020.

[118] Michael H Veatch. *Linear and Convex Optimization: A Mathematical Approach*. John Wiley & Sons, 2020.

[119] Luke N Veldt. *Optimization Frameworks for Graph Clustering*. PhD thesis, Purdue University Graduate School, 2019.

[120] Nate Veldt, David Gleich, and Anthony Wirth. Learning resolution parameters for graph clustering. In *The World Wide Web Conference*, pages 1909–1919, 2019.

[121] Hélene Verhaeghe, Siegfried Nijssen, Gilles Pesant, Claude-Guy Quimper, and Pierre Schaus. Learning optimal decision trees using constraint programming. *Constraints*, 25(3):226–250, 2020.

[122] Natalia Vesselinova, Rebecca Steinert, Daniel F Perez-Ramirez, and Magnus Boman. Learning combinatorial optimization on graphs: A survey with applications to networking. *IEEE Access*, 8:120388–120416, 2020.

[123] Linchuan Wei, Andrés Gómez, and Simge Küçükyavuz. On the convexification of constrained quadratic optimization problems with indicator variables. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 433–447. Springer, 2020.

[124] Judy Whitehead. *Microeconomics: a global text*. Routledge, 2014.

[125] Angelika Wiegele. Biq mac library—a collection of max-cut and quadratic 0-1 programming instances of medium size. *Preprint*, 51, 2007.

[126] David P Williamson. *Network flow algorithms*. Cambridge University Press, 2019.

[127] Gerhard J Woeginger. Exact algorithms for np-hard problems: A survey. In *Combinatorial optimization—eureka, you shrink!*, pages 185–207. Springer, 2003.

[128] Henry Wolkowicz, Romesh Saigal, and Lieven Vandenberghe. *Handbook of semidefinite programming: theory, algorithms, and applications*, volume 27. Springer Science & Business Media, 2012.

[129] Lidong Wu, Wen Xu, Ding-Zhu Du, and Zhao Zhang. *Combinatorial optimization and applications*. Springer, 2011.

[130] Gal Yehuda, Moshe Gabel, and Assaf Schuster. It's not what machines can learn, it's what we cannot teach. In *International Conference on Machine Learning*, pages 10831–10841. PMLR, 2020.

[131] Shui Yu, Meng Liu, Wanchun Dou, Xiting Liu, and Sanming Zhou. Networking for big data: A survey. *IEEE Communications Surveys & Tutorials*, 19(1):531–549, 2016.

[132] Fuzhen Zhang. Positive semidefinite matrices. In *Matrix theory*, pages 199–252. Springer, 2011.

الخلاصة

تركز هذه الأطروحة على حل مشكلة NP-hard (غير الحتمية متعددة الحدود) باستخدام متغير واسع النطاق. حيث أن مشكلة K-CLUSTER (البحث عن الرسم البياني الفرعي الأكثر كثافة لحجم ثابت k) بحدود شبه غير محددة هي مشكلة NP-hard الكلاسيكية للتحسين التوافقي وقد ثبت أنها مشكلة NP-Hardness في تجميع الرسم البياني . تولد الاستراتيجية علاقة جيدة مع القيمة المثلى للمشكلة الأصلية. تم تطوير طريقة لاغرانج المعزز كبديل لطريقة penalty ، وتم تطوير تقنية هجينة تتغير بين الطريقتين بناءً على قيمة المعلمة. تم اختبار النموذج الجديد باستخدام العديد من الرسوم البيانية من مكتبة Biq Mac، حيث تم استخدام لغة جوليا للحصول على النتائج. يتفوق نهجنا الجديد على أساليب penalty من حيث السرعة والمتانة. أخيرًا ، أظهرت خصائص التقارب النظرية للخوارزمية المقترحة أنها التقنية المفضلة لحل توسيع البرمجة شبه المحددة للمسائل ثنائية التفرع التربيعية.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية التربية للعلوم الصرفة
قسم الرياضيات

الطرق العددية المثلى لحل البرمجة شبه المحددة لمسائل NP-Hardness

أطروحة مقدمة الى مجلس كلية التربية للعلوم الصرفة في جامعة بابل
كجزء من متطلبات نيل درجة الدكتوراه فلسفة في التربية / الرياضيات

من قبل
احمد حسن حميد محمود
بإشراف
أ. م. د احمد صباح احمد الجيلاوي

1443 هـ 2022 م