Model and Solutions to Campus Parking Space Allocation Problem

By

Luke Oluwaseye Joel

(211559858)

Submitted in fulfilment of the academic requirements
for the degree of Master of Science
in the School of Computer Science
University of Kwazulu-Natal
Westville Campus, South Africa.

Supervisor

Dr. A. O. Adewumi

June, 2013

COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE

DECLARATION

The work described by this thesis was carried out at the University of Kwazulu-Natal, School of Mathematics, Statistics and Computer Science, University of Kwazulu-Natal, Westville Campus, under the supervision of Dr. Adewumi A. O.

This thesis is entirely, unless specifically contradicted in the text, the work of the candidate and has not been previously submitted, in whole or in part, to any other tertiary institution. Where use has been made of the work of others, it is duly acknowledged in the text.

Signed: _____ Student Name: _____

As the candidate's supervisor I have/have not approved this dissertation for submission.

Signed: _____ Supervisor Name: _____

COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE

DECLARATION - PLAGIARISM

I, ..., declare that

1. The research reported in this dissertation, except where otherwise indicated, is my original research.
2. This dissertation has not been submitted for any degree or examination at any other university.
3. This dissertation does not contain other persons data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
4. This dissertation does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
   (a) Their words have been re-written but the general information attributed to them has been referenced
   (b) Where their exact words have been used, then their writing has been placed in italics and inside quotation marks, and referenced.
5. This dissertation does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the dissertation and in the References sections.

Signed

.....

Declaration Plagiarism 22/05/08 FHDR Approved

Dedication

This thesis is dedicated to the members of my family in memory of our loving and caring mother.

Acknowledgements

My ultimate thanks, praise, and honour go to the Omniscient, Omnipotent and Omnipresent God who saw me through this study.

I owe gratitude to my supervisor, Dr. Adewumi Aderemi, for his guidance, support and patience towards me during my entire period of study. I am also grateful to Dr. B. A. Sawyerr and Dr. Blamah for their programming assistance, and the optimization and modelling research group members for their moral support and contributions.

I offer my profound gratitude to my family members for their spiritual, financial and moral support, may the Almighty God bless you abundantly.

I am thankful to the following people: the technical personnel- Mr Greenwood and Mr Jay, Mrs Chairmaine, Mrs Moodley and Mrs Sheeren of the School of Mathematical, Statistics and Computer Science.

A million thanks to the Deeper Life Bible Church leaders and workers in Westville, Howard, and Edgewood campuses for their prayers. I also want to express my gratitude to my dear colleagues in the masters' LAN, and all my friends far and near.

Finally, I am indebted to every other person not mentioned here who might have in one way or the other contributed to the success of this project and my overall stay in South Africa during my masters' study. May God in his infinite mercy bless you all abundantly (Amen).

Abstract

Parking is considered a major land use challenge in campus planning. The problem can be in terms of scarcity (few available spaces compared to demand) or management (inefficient usage of available facilities). Many studies have looked at the parking problem from the administrative and management points of view. However, it is believed that mathematical models and optimization can provide substantial solution to the parking problem. This study investigates a model for allocating car parking spaces in the university environment and improves on the constraints to address the reserved parking policy on campus. An investigation of both the exact and heuristic techniques was undergone to provide solutions to this model with a case study of the University of KwaZulu-Natal (UKZN), Westville Campus.

The optimization model was tested with four different set of data that were generated to mimic real life situations of parking supply and demand on campus for reserved and unreserved parking spaces. These datasets consist of the number of parking lots and office buildings in the case study. The study also investigate some optimization algorithms that can be used to obtain solutions to this problem. An exact solution of the model was generated with CPLEX solver (as incorporated in AIMMS software). Further investigation of the performance of the three meta-heuristics to solve this problem was done. A comparative study of the performance of these techniques was conducted. Results obtained from the meta-heuristic algorithms indicate that the algorithms used can successfully solve the parking allocation problem and can give solutions that are near optimal. The parking allocation and fitness value for each of the meta-heuristic algorithms on the sets of data used were obtained and compared to each other and also to the ones obtained from CPLEX solver. The results suggest that PSwarm performs better and faster than the other two algorithms and gives solutions that are close to the exact solutions obtained from CPLEX solver.

Contents

List of Figures

List of Tables

List of Abbreviations

AIMMS	Advanced interactive multidimensional modeling system
AP	Assignment problem
BB	Branch and bound
CPSA	Campus parking space allocation
COP	Combinatorial optimization problem
GA	Genetic algorithm
GPS	Generalised pattern search
IPM	Interior point method
KP	Knapsack problem
LP	Linear programming
MATLAB	Matrix laboratory
PS	Pattern search
PSA	Parking space allocation
PSO	Particle swarm optimization
PSwarm	Particle swarm pattern search
SA	Simulated annealing
SAP	Space allocation problem
TS	Tabu search
TSP	Travelling salesman problem

Outcome of Research Work - Publication

Details of publication that form part and/or include research presented in this thesis.

- Joel L. O., Adewumi A. O. and Sawyerr B. A.(2013). An Exact Solution for Allocating Car Parking Spaces on Campus. *Proceedings of the International Science, Technology, Education, Arts, Management and Social Science (iSTEAMS) Research Nexus Conference.* University of Ibadan, Ibadan, Nigeria, May 30 - June 1(pp 463-472)

  See Appendix E

Chapter 1

Introduction

> "I have sometimes thought of the modern university as a series of individual faculty entrepreneurs held together by a common grievance over parking ", [21].

1.1 Background of Study

Parking is an act of manoeuvring a vehicle into a location where it can be left temporarily. It is a major concern in transportation planning and traffic management of organisations all over the world. The problem of parking is more pronounced in urban areas [69] especially in closed communities like the university environment [58]. Parking poses a problem due to the limited supply of parking spaces compared to demand [2] and in terms of the ineffective usage of parking facilities. Experts estimate that 30 percent of traffic in the city is caused by people driving around looking for parking [49, 17]. IBM [49, 50] conducted a survey of commuters in twenty international cities, Johannesburg (South Africa) inclusive, on six continents, and found that about six out of every ten drivers gave up their hunt for a parking space at one time or the other. It was also discovered that, drivers, in pursuit of a desired parking space, spent about 20 minutes searching. A year-long study that was also conducted [49] found that drivers in some of the districts in Los Angeles drove in excess of 950,000 miles, produced 730 tons of carbon-dioxide, and used 47,000 gallons of gas in search of parking spaces.

This problem of finding a suitable parking space now plagues university environments irrespective of where the university is located [2]. Parking is a big problem on campuses because of the demand for parking by the faculty, staff, and a large numbers of students. This three-fold demand for parking has brought about an acute shortfall of parking spaces and has become concern for university planners [2]. Hence, an optimal solution to the allocation of these limited parking spaces on campus needs to be investigated.

Figure 1.1: Parallel parking of cars [123]

Parking space is essentially influenced not only by the limited amount of space and the demand for such but also by the way available spaces are organized and/or arranged. This determines the number of vehicles that can be parked in a given space [37]. We discuss briefly below some types of parking as presented in literature and some restrictions that are often put on them due to this limited available spaces.

1.1.1 Types of Parking

- Parallel parking: Parked cars, in parallel parking, Figure 1.1, are arranged in a line [123]. This is most common on the street-side (also called on-street) parking where the front bumper of one car is facing the back bumper of another adjacent car. Parallel parking works well in extremely narrow, linear spaces but it is difficult to manoeuvre for most drivers and is an inefficient use of on-street space [37, 123].

- Perpendicular parking: In perpendicular parking, Figure 1.2, cars are parked at a 90° angle (perpendicularly) to an aisle, curb or wall. This type of parking is common in an off-street parking lot where the driver can either park with a back-in or head-in parking style [37, 124]. The perpendicular parking handles most vehicles per square meter of pavement and works well with either one- or two-way aisles [37, 124].

- Angle parking/echelon parking: Cars can also park at an angle less than 90°, which could be 30°, 45° and 60°, to an aisle, curb or wall. This type of parking, Figure 1.3, is known as angle parking or echelon parking. Although this type of parking is common in an off-street parking lot, it could also be used for on-street parking [37, 122]. As with perpendicular parking, parking in an angle could be done in a head-in configuration (most common) or back-in style. This mode of parking is known for its ease of manoeuvring in and out of parking spaces with good visibility to the rear though it does not work well with two-way aisles [37, 122]and it requires more pavements per vehicle than the perpendicular parking configuration.

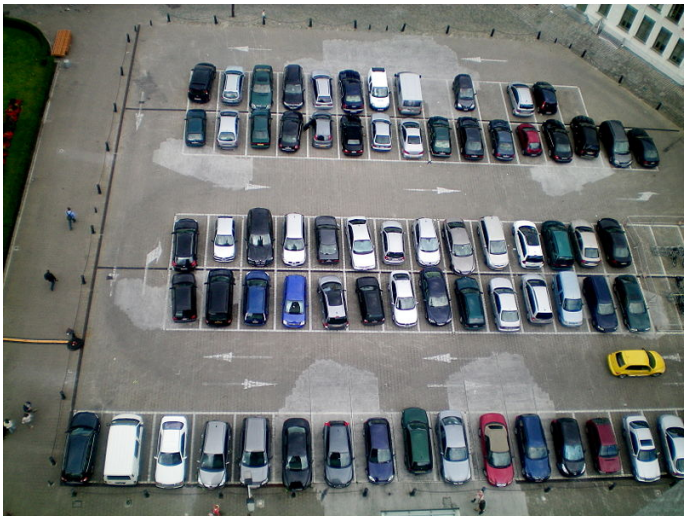

Figure 1.2: Perpendicular parking of cars [124]

Figure 1.3: Cars showing angle parking [123]

1.1.2 Parking Restrictions

Parking restrictions are the rules used in regulating parking activities. Parking is often with restrictions due to the few available parking spaces compared to demand for such spaces. For example, drivers could be allowed to park on the side of a street, or park in a location for a limited period of time after which a certain amount of money would be paid. Following are some of the parking restrictions [81].

- **Smart Parking Meter:** Payment for parking could be used to restrict both on- and off-street parking spots [81]. The use of parking meter, smart parking meter or parking pay station could be employed for parking payments. The use of these devices help in money collection from customers. Parking meter serves on parking space per time while parking pay station serves multiple parking spaces per time. On the other hand, smart parking meter gives more payment options to customers other than cash payments.

- **Time limits:** Parking restrictions could also be enforced when drivers are allowed to park for only a certain period of time [81]. In some circumstances, payments for parking are only done within specific hours of the day, while the other hours are free. The time limit restriction also determines whether the parking will be for a short or long period of time. Disk parking allows for monitoring of the length of time a car has been parked.

- **Permit:** A permit is a licence to park in a given parking space. It could be a decal, hangtag, or Radio-Frequency Identification (RFID) [81] showing that only people having these parking permit can park in the designated areas.

1.2 Statement of Problem

The Parking Space Allocation (PSA) problem is concerned with the distribution of available parking spaces to a set of people in order to minimize parking space misuse and to minimize the distance walked by each user from the parking lot to their destination. The PSA problem affects the lives of almost everyone who has a car in the society especially in the campus environment. The problem could be in the time taken to get a parking space, the tedious effort exerted in getting a permit or in the distance between the parking space and the desired destination.

The former President of the University of California, Clark Kerr once said [21] “I have sometimes thought of the modern university as a series of individual faculty entrepreneurs held together by a common grievance over parking”. It is this “grievance over parking”in the university campuses that this study intends to investigate. Many studies have looked at the CPSA problem from the administrative and management point of view, however this study will examine the problem from an optimization point of view. The study addresses this problem of parking allocation in

the university environment by formulating an appropriate model of the problem which caters for both reserved and unreserved parking policies in the campus environment and then applies three different meta-heuristic algorithms for its solution.

1.3 Motivation

Parking is a vital part of transportation since vehicles have to be parked most of the time during the day [109, 39, 6], hence it is essential to get a good management of parking spaces or optimize the use of it. Parking problems, among other things, are major problems facing the society, especially the university environment due to the cost of parking facilities and the limited number of available parking spaces. Narragon et al. [7] explained that although building more parking facilities appears to be the direct solution to the parking problem, it is difficult to do so due to space and fund challenges. This only gives one an option of ensuring that the existing facilities are effectively and efficiently utilized through policies that encourage such. Some compelling reasons [109] to ensure efficient use of parking facilities are:

- It saves the parking managers or officials the cost of building more parking areas.
- It gives the users opportunity to explore other alternatives.
- It helps in having a more attractive and less congested environment since proper and effectively managed parking leaves space in the environment for other use.
- It aids in revenue generation which could be used for better management of parking facilities or for other purposes.
- It enables management to explore technologies and/or innovations that could be used in the optimization of parking resources.

1.4 Objective of the Study

The objectives of this study are:

1. To study the existing CPSA problem.
2. To formulate the problem as a Combinatorial Optimization Problem(COP).
3. To develop an appropriate model for solving the CPSA problem.
4. To apply heuristic techniques such as Genetic Algorithm (GA), Pattern Search(PS), and Particle Swarm Pattern Search(PSwarm) to solve the problem.
5. To compare the results in (4) with that obtained from CPLEX software.

6. To analyse the performance of each of the techniques

7. To draw valuable conclusion(s) from the results obtained.

1.5 Definition of Terms

Here are some related words and definitions.

- **Allocation** - Refers to an authorization for a specific parking assignment [22, 90].
- **Campus** - The buildings of a college or university and the land that surrounds them [22, 23].
- **Building** - A physical location to which a user goes after parking his car in the parking lot [22, 57].
- **Model** - A simplified representation of a complex real-world event or structure [22, 8].
- **Optimal solution** - Alternative approach that best fits the situation, employs resources in the most effective and efficient manner, and yields the highest possible return under the given circumstances [22, 110].
- **Optimization** - Finding an alternative with the most cost effective or highest achievable performance with the given constraints, by maximizing desired factors and minimizing undesirable ones [22, 8].
- **Parking lot** - A dedicated area that is intended for parking vehicles, having more than one parking space [22, 124].
- **Parking Space** - A space indicated between two white lines where a vehicle or motorcycle can park [22, 57].
- **Parking permit** - An official document which gives permission to a car owner to park in a designated parking space [22, 58].
- **Reserved space** - A parking space designated for use by a particular user. A reserved space is not meant to be shared with any other user [22, 84].
- **Staff** - The personnel employed by the University who assists in performing the business of the university [22, 21].
- **Unreserved space** - A parking space designated for use by many users [22, 58].
- **User** - Anyone who is in need of a parking space [22, 21]. The term parking user could also be used.

- **Visitor** - Any person who is NOT classified as a member of faculty staff, student, or vendor at the University and will be at the university for short periods of time to conduct business [22, 21]. If staying overnight, the visitor is then considered a guest and must be sponsored by a student, staff or employee member.

1.6 Thesis Overview

The rest of the Thesis is organised as follows: An introduction to optimization, some similar examples of COP, classifications of optimization techniques used in addressing optimization problems and the penalty method used for constrained optimization problems are discussed in **Chapter 2**. An introduction to the campus space allocation problem with a review of related works, the formulated optimization model for the allocation problem, the solution idea employed in the study, and the description of the exact optimization solver as well as the meta-heuristic algorithms used in obtaining solutions to the problem are presented in **Chapter 3**. The dataset, experimental settings and the results obtained from the exact solver and the meta-heuristic algorithms are discussed in **Chapter 4**. An overall conclusion and suggestion of areas for future research is given in **Chapter 5**.

Chapter 2

Background and Related Works

> "The essence of an optimization problem is: Catching a black cat in a dark room in minimal time. A constrained optimization problem corresponds to a room full of furniture ", [12].

2.1 Introduction to Optimization

Optimization is a process of making a system or decision as functional or effective as possible. Due to the significance of optimization, especially global optimization, its application is often used in daily life. A desire to achieve optimality is one of the most underlying precepts in the world today [110, 129]. Optimization problems can be found in the biological sciences, engineering, economics, business, architecture, computer science, management and physical sciences. Optimization, also known as mathematical programming, aids in finding solutions that give maximum performance, profit, output or happiness; or minimum cost, waste or unhappiness [43, 129]. It solves the problem of deciding how to optimally allocate scarce resources such as people, materials, spaces, money and land.

An optimization model attempts to optimize an objective function by searching for the best set of decision variables that obey the given constraints [71, 110, 67]. It is common to use the word "optimize ", which means to maximize or minimize, in any optimization problem. A mathematical function, also known as an objective function, to be optimized could be a function of only one variable, called single-objective problem, or a function of multiple variables, called multi-objective problem. Furthermore, an objective function could also be constrained or unconstrained [8].
There are three essential components [8, 67] of an optimization model which are:

1. The objective function or cost function

2. The decision variables

3. The set of constraints to be satisfied

Optimization has practical applications in many fields of study and human endeavour including natural sciences, engineering and social sciences. Hence, researchers across these fields often use an optimization model for better presentation of the problem of interest. The structure of an optimization model can be represented as follows:

$$Maximize \qquad f(x) \tag{2.1}$$

subject to

$$g_i(x) \leq G_i \qquad where \;\; i = 1, 2, \ldots, m \tag{2.2}$$

$$h_j(x) = H_j \qquad where \;\; j = 1, 2, \ldots, n \tag{2.3}$$

$$x \geq 0 \tag{2.4}$$

where $f(x)$ is the objective function of a single variable x; $g_i(x)$ and $h_j(x)$ are the constraint functions of the variable $x \in R^n$. G_i and H_j are constants. A non-negative constraint, $x \geq 0$, is important for most real-world problems since they often deal with either maximizing profits or minimizing costs which would be meaningless if they are negative. Given that x represents a vector of variables, where $\omega = (x_1, x_2, ..., x_k)$, the above model can be rewritten for multiple variables as follows:

$$Maximize \qquad f(\omega) \tag{2.5}$$

subject to

$$g_i(\omega) \leq G_i \qquad where \;\; i = 1, 2, \ldots, m \tag{2.6}$$

$$h_j(\omega) = H_j \qquad where \;\; j = 1, 2, \ldots, n \tag{2.7}$$

$$\omega \geq 0 \tag{2.8}$$

Optimization follows a step by step procedure in order to solve a given problem. There are different types of a given optimization problem. Hence, we discuss the process of optimizing a given problem below and the different types of optimization problems as given in literature.

2.1.1 Optimization Process

The optimization process helps decision-makers to determine realistic and practical outcomes to complex management problems [8]. The process of optimization is shown in Figure 2.1. The optimization process begins by defining a real-world problem to extract some details and relevant information necessary to create a mathematical model [114]. Then, an optimization technique or method would be applied to the model with the aid of computer programs. From Figure 2.1, the first process is **analysis**, as shown by the arrow moving from the real-world problem box to the algorithm, model and solution techniques box. Here, the task of extracting relevant details that is essential for building an efficient mathematical model takes place. The process of analysis is important because it lays bare crucial elements of the problem and it gives insights as to how to address the problem [114].

The second process, as shown by the arrow moving from the algorithm, model and solution technique box to the computer implementation box, is the use of **optimization methods**. The knowledge of these optimization methods and their implementation procedure is essential in order to obtain solution for the mathematical model. In order to be sure that the optimization method is performing as expected, the **verification** process is used. The verification process is shown by the arrow moving from the computer implementation box back to the algorithm, model and solution technique box.

The optimization process is completed by moving from the algorithm, model, or solution technique box to the real world problem box. This process is called **validation and sensitivity analysis**. Here, the results obtained are compared with the real world situation. This process determines whether the results are appropriate and whether there is need to modify the model or chose another solution technique [114].

2.1.2 Classification of Optimization Problems

As earlier explained, optimization problems seek to find the best values of a certain number of variables with a given set of constraints. Hence, an optimization problem can take different forms and classifications [129]. The classification, given by Sarker et al [8] as seen in Figure 2.2 is based on:

- **Objective function**: The optimization problem based on an objective function can either be a single objective optimization problem or a multiple objective optimization problem
- **Constraints**: An unconstrained optimization problem only has the objective function to be optimized without any set of constraints that have to be satisfied while the constrained optimization problem must satisfy some set of equality and/or inequality constraints. Literature reveals that the study of unconstrained optimization problem is essential, though most real world optimization problems are constrained.

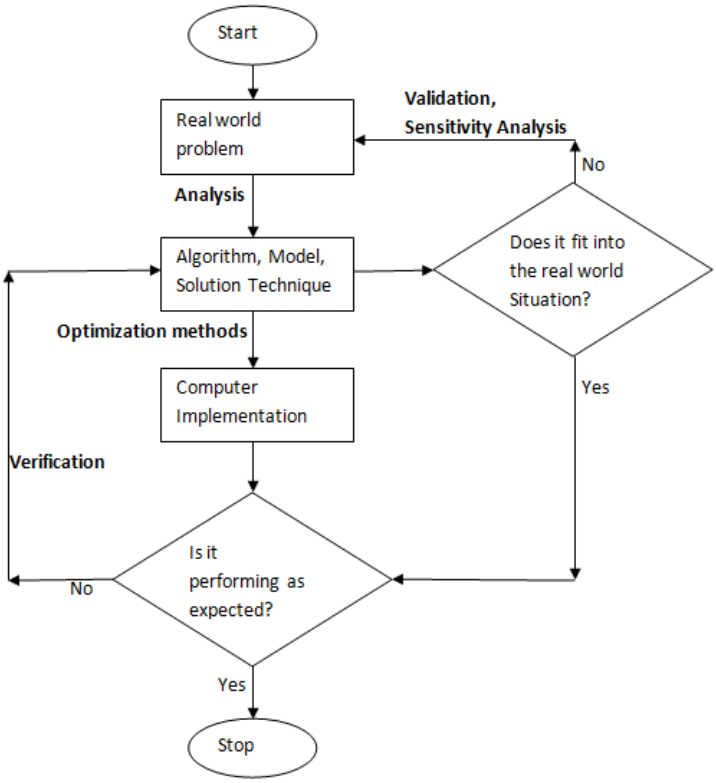

Figure 2.1: The optimization process [114]

- **Variable**: An optimization problem can only have integer or discrete variable values, known as COP; a real variable value, known as continuous optimization problem; or a combination of integer and continuous variable values, known as mixed optimization problem.
- **Function**: The classification of optimization problems based on function are: linear optimization problem or non-linear optimization problem; convex optimization problem or non convex optimization problem; differentiable (those with derivative) optimization problem or non-differentiable (those without derivative) optimization problem.

Some common types of optimization problems are unconstrained non-linear optimization problems, constrained convex optimization problems, integer optimization problems (which is generally known as COP) etc.

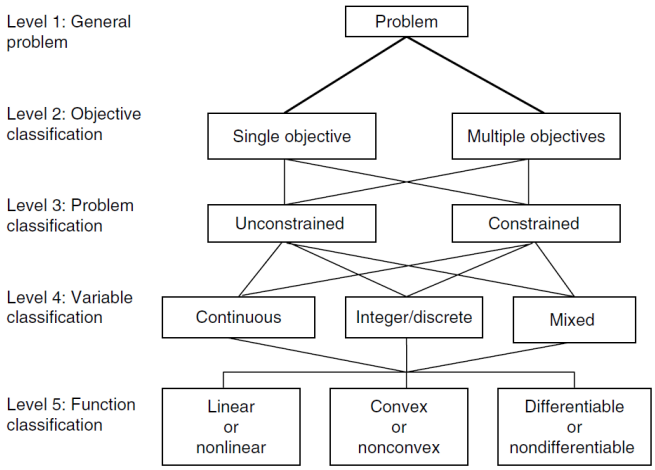

Figure 2.2: Classification of optimization problems [8]

2.2 Combinatorial Optimization Problem

COP is a subset of the general optimization problem in which its set of decision variables take only discrete or integer values [13, 8, 59, 71]. It helps in the efficient allocation of limited resources when the values of the variables are restricted to integral values. Gen and Cheng [71] classified the characteristics of COP as:

1. Permutation of some set of items subject to some constraints. Examples are vehicle routing problem or scheduling problem

2. Combination of some set of items subject to some constraints. Examples are set-covering or grouping problem

3. Both permutation and combination of some set of items subject to some constraints. An example is parallel machine scheduling problem

Hoffman and Padberg in [59] note that "The versatility of the combinatorial optimization model stems from the fact that in many practical problems, activities and resources, such as machines, airplanes and people, are indivisible [59]."Examples of COP include Knapsack Problem(KP), Travelling Salesman Problem(TSP), Assignment Problem(AP), and Space Allocation Problem(SAP). These are discussed below:

2.2.1 Knapsack Problem

KP is one of the NP-hard COPs that often appear in resource allocation with financial constraints. It determines the items to include in a collection of items so that the total weight does not exceed the given limit and the total value of the items taken is as large as possible. The integer program formulation of the KP with the decision variable

$$x_i = \begin{cases} 1 & \text{if item } i \text{ is picked} \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

is as follows:

$$Maximize \quad \sum_{i=1}^{n} p_i x_i \tag{2.10}$$

subject to

$$\sum_{i=1}^{n} a_i x_i \leq K \tag{2.11}$$

$$x_i \epsilon \{0, 1\} \quad \forall \ i = 1, 2, ..., n$$

where $i = 1, ..., n$ is the number of items to be chosen, a_i is the weight, p_i is the profit attached to choosing each item, and K is the capacity of the knapsack.
Common variations of KP are:

- **0-1 Knapsack problem**: This is the most common type of KP which restricts the number of copies of each kind of item that is taken to be zero(0) or one(1).
- **Bounded knapsack problem**: This restricts the number of copies of each kind of item to be taken to a minimum integer value (say C_i)
- **Unbounded knapsack problem**: This type of KP puts no restriction on the number of copies of each kind of item to be taken. It is also called the integer knapsack problem.
- **Multiple-choice knapsack problem**: This is another variant of a knapsack problem in which the items are subdivided into classes and exactly one item must be taken from these classes.

More information on KP can be found in [76, 66, 75].

2.2.2 Travelling Salesman Problem

This is an NP-Hard COP. It is a benchmark problem for many optimization methods. It determines the shortest possible route that connects a given list of cities with their pairwise distances exactly once. TSP is a minimization problem which can be modelled as an undirected weighted graph, such that cities are the graph's vertices, paths are the graph's edges, and a path's distance is the edge's length.

The mathematical formulation for the TSP is given as:

$$Minimize \quad Z = \sum_{i}^{N} \sum_{j}^{N} C_{ij} x_{ij} \tag{2.12}$$

subject to

$$\sum_{i=1}^{N} x_{ij} = 1 \quad \text{for } j\epsilon\{2,3,...,N\} \tag{2.13}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \quad \text{for } i\epsilon\{2,3,...,N\} \tag{2.14}$$

$$\sum_{i\epsilon S}\sum_{j\epsilon S} x_{ij} \leq |S| - 1 \quad \text{for } S \subset V \tag{2.15}$$

$$x_{ij}\epsilon\{0,1\} \quad \text{for } i,j\epsilon\{2,3,...,N\}$$

$|S|$ denotes the number of elements in the subset S , $V = \{1,2,...,N\}$, N is the total number of cities, C_{ij} is the cost of travelling from node i to node j, and x_{ij} is the decision variable such that

$$x_{ij} = \begin{cases} 1 & \text{if the salesman travels from node } i \text{ to node } j \\ 0 & \text{otherwise} \end{cases} \tag{2.16}$$

The objective function in Equation (2.12) is used to minimize the total distance travelled. The constraints in Equation (2.13) and (2.14) ensure that each node must be visited exactly once. And the constraint in Equation (2.15) eliminates the sub-tours. Some relevant literature on the TSP can be found in [63, 56, 65, 101].

2.2.3 Assignment Problem

The assignment optimization problem seeks to minimize the total cost of assigning a number of agents to some given tasks. A more applicable model [42] assigns several tasks to a single agent; if and only if, these tasks assigned to the agent do not require more than the available resources. The mathematical formulation of an assignment problem is given as:

$$Minimize \quad \sum_{i}^{m}\sum_{j}^{n} C_{ij}x_{ij} \tag{2.17}$$

subject to

$$\sum_{j}^{n} r_{ij}x_{ij} \leq b_i \quad \forall \ i = 1,2,...,m \tag{2.18}$$

$$\sum_{i}^{m} x_{ij} = 1 \quad \forall \ j = 1,2,...,n \tag{2.19}$$

where,

$i = 1,2,...,m$ is the set of agent indices

$j = 1,2,...,n$ is the set of task indices

C_{ij} is the cost of assigning an agent i to task j

r_{ij} is the resource required by agent i to do task j

b_i is the resource available to agent i

and

$$x_{ij} = \begin{cases} 1 & \text{if agent } i \text{ is assigned to task } j \\ 0 & \text{otherwise} \end{cases} \tag{2.20}$$

A common type of this problem is the Linear Assignment Problem(LAP) which occurs when the number of agents equals the number of tasks to be performed and the total cost of the assignment for all the tasks equals the sum of the costs for each agent. Some relevant literature for the assignment problem can be found in [97, 68, 38, 20].

2.2.4 Space Allocation Problem

SAP deals with the distribution of limited spaces among a certain number of entities with the goal of finding the optimal use of the space while satisfying some constraints [31, 55]. The space allocation problem basically has two objectives: Firstly, to minimize the misuse of the space so that entities are allocated to spaces efficiently. Secondly, to minimize the penalty for the violation of soft constraints in the problem. There are two types of constraints: the hard constraint and the soft constraints. The hard constraint is required to obtain a feasible solution while the soft constraint does not have to be satisfied, hence, it should be penalized.

The mathematical formulation of the space allocation problem is given as:

$$Minimize \quad \sum_{i}^{n} \sum_{j}^{m} C_{ij} X_{ij} \tag{2.21}$$

subject to

$$\sum_{i=1}^{n} X_{ij} \leq T_j \quad \forall \ j = 1, ..., m \tag{2.22}$$

$$L_i \leq \sum_{j}^{m} X_{ij} \leq U_i \quad \forall \ i = 1, ..., n \tag{2.23}$$

where,

$i = 1, ..., m$ is the set of entity indices

$j = 1, ..., n$ is the set of space area indices

C_{ij} is the cost of assigning an entity i to a space area j

T_j is the total space capacity

L_i and U_i are the lower and upper bounds of the number of entities to be allocated

Equation (2.22) indicates that the total number of allocations must not exceed the space area capacity. Equation (2.23) represents the constraints of the lower and upper bounds.
SAPs are NP-hard in nature and similar to scheduling problems. Instances of SAP in real life are Shelf Space Allocation Problem (SSAP) [31, 26], Office Space Allocation Problem (OSAP) [92, 93] and Hostel Space Allocation Problem (HSAP) [82, 31]. The CSPA problem considered in this study is also an instance of the SAP. Details of this will be discussed in chapter 3.

2.3 Optimization Techniques

Optimization problems can be dated to the history of man. Humans tend to always search for optimal solutions to problems encountered, this has led to a search for techniques and methods to solve optimization problems. Before the twentieth century [129], a number of scientists and researchers had been able to explore some techniques in solving many optimization problems. For example, Johannes Kepler solved the so-called "marriage problem "or "secretary problem "to optimality in 1613. The problem was later developed into the field of probability optimization by its formal introduction by Martin Gardner in 1960; Sir Isaac Newton also solved the problem of minimal resistance of the body shape in 1685, which become the problem of calculus of variations. Gaspard Monge looked into the transportation problem with known initial and final spatial distribution; L. A. Cauchy in 1847 suggested a general iterative method for solving systems of equations which are now known as the gradient method and the steepest descent method.

However, at the turn of the twentieth century [129], the use of optimization models and optimization techniques became more obvious. A few milestones are: J. Jensen in 1906 introduced the concept of convexity (a basis for convex optimization); L. Kantorovich in 1939 developed an algorithm for linear programming; George Dantzig in 1947 invented the simplex method for solving large-scale linear programming problems; Harold Kuhn and A. W. Tucker in 1951 studied the nonlinear optimization problem and re-developed the optimality condition (a necessary condition in nonlinear programming for a solution to be optimal). Finally before the 1960s explosion of literature on optimization, Richard Bellman in 1957 developed the dynamic programming. Specifically, after 1960, the field of optimization was further expanded with the introduction of more literature on optimization.

Since there are different optimization problems, there are also different optimization techniques that could be used to address these problems, this is because certain techniques are more suitable for some types of optimization problems than others[129]. The techniques for solving these problems can be classified into two important groups: the exact (classical) optimization techniques and the heuristic/meta-heuristic optimization techniques (See Figure 2.3 for an overview of these techniques/methods). Some common exact techniques used by most optimization solvers and in

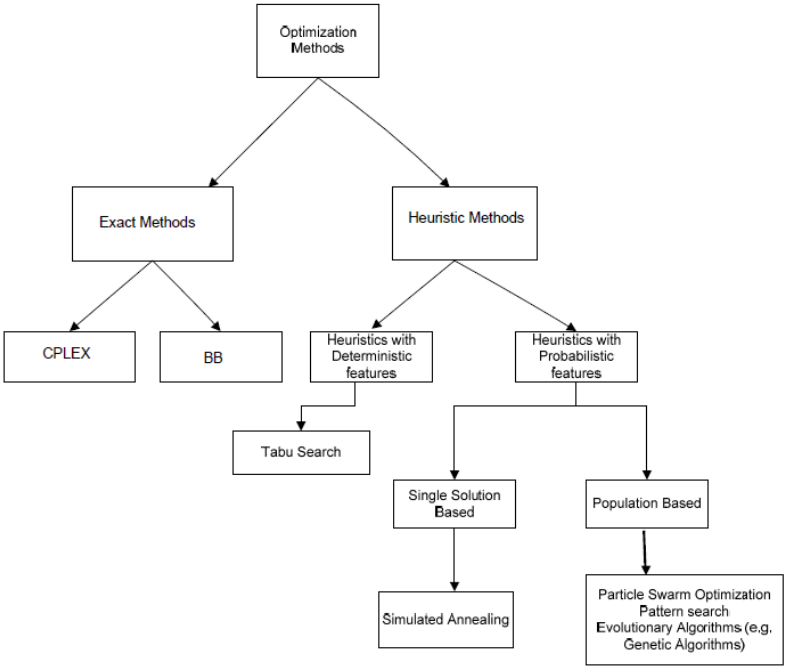

Figure 2.3: Classification of optimization methods [82]

literature will be discussed along with the meta-heuristic techniques used in this study.

2.3.1 Exact Optimization Techniques

The exact or classical optimization methods are useful in finding the optimum solution to a given optimization problem. These methods obtain their optimum solution in an iterative manner by starting from an initial solution. Three of such methods that are commonly used to provide solution to real world optimization problems are discussed.

2.3.1.1 Simplex Algorithm

The simplex algorithm is a popular algorithm in optimization techniques for linear programming [125, 8]. The algorithm was developed by George B. Dantzig in 1947. The algorithm is used to determine the feasible solution space, which is the space bounded by the constraints and variables bounds, of a given linear programming problem. This then helps in identifying the optimal point from the solution space. The simplex method uses a search process [8] that evaluates solutions obtained from the intersections of constraint equations, called corner points, to find the optimal corner points through the boundary of feasible space. The search process often starts at the origin and then moves to an adjacent corner point that gives a minimum (for minimization problem) objective function value. The process continues until there are no further improvements. Sometimes, the number of the corner points might be an exponential function of the problem dimension which

would necessitate being visited by the simplex method. The basic steps of the algorithm can be expressed briefly as follows [8]:

1. Standardize the problem into an LP tableau.
2. Generate an initial feasible solution, called a basis.
3. Test the solution for optimality.
    - If not optimal, improve it (go to Step 4);
    - Otherwise go to Step 6.
4. Generate an improved solution by identifying the leaving and entering variables to the basis and updating the tableau.
5. Check for optimality (as in Step 3).
    - If not optimal, repeat Steps 4 and 5.
    - If optimal, go to Step 6.
6. Stop.

2.3.1.2 Interior Point Algorithm

Interior Point Method (IPM) is another classical optimization technique that can be used to solve linear and nonlinear convex optimization problems. The method, which has its main idea to obtain the optimal solution via the interior of the feasible space [72, 105], was proposed by Narendra Karmarkar in 1984. It is also known as a class of barrier methods. Its variants are being used to obtain solution to many real world optimization problems. There are three main classifications [72] of IPM, which are projective methods, affine-scaling methods and primal-dual methods. The popular and most efficient one among these classifications is the primal-dual method. The following show some advantages of IPM over the simplex method: It is not affected by the problem of degeneracies and it uses less number of iterations for larger LP problems than the simplex method [72, 8]. However, the IPM is not able to detect a possible infeasible problem and each of its iteration is computationally more expensive than the simplex method. The outline [72] of the IPM is as follows:

1. Transform the inequality constraints in the given problem to equality constraints by introducing slack variables
2. Treat the non-negativity conditions by appending the logarithmic barrier functions to the objective function
3. Choose a proper starting point such that the non-negativity conditions are satisfied.

4. Compute the barrier parameter that was introduced in step 2
5. Solve the system of equations by some iterative methods
6. Determine the step size that preserves the non-negativity condition and update the solution.
7. If the solution meets the convergence criterion, optimal solution is found, otherwise go back to step 4.

2.3.1.3 Branch and Bound Algorithm

Branch and Bound(BB) method is one of the efficient methods used to solve discrete and COPs [89]. It uses partial enumeration, since total enumeration in most real world COPs are computationally infeasible [89, 8], to explore its solutions and for finding the optimal solution to a given problem. The method uses a tree structure with nodes and edges to represent its partial solutions. A partial solution has some fixed discrete variables and others are free. The partial solutions on the same level are compared with each another and the minimum of solution is picked. Then, the process continues vertically downwards until all the partial solutions are examined which often leads to the discovery of the overall optimum solution. Each node or partial solution is expanded to the end of it (called fathoming). Those nodes that are good enough are investigated further (called branching), while those that are not good enough to be investigated further due to the bounding conditions are terminated (called pruning). The BB method was proposed in 1960 by A.H. Land and A.G. Doig [120]. The Branch and Bound algorithm [8] is explained as follows:

1. Solve the problem as a continuous relaxation problem
2. Use the solution obtained to compute the bound conditions for the node. The continuous solution obtained will always be better than the integer solution
3. The bound condition in step 2 is used to branch the original problem into two sub-problems
4. Each of the sub-problems are solved as a continuous relaxation problem
5. If its optimum solution satisfies the integrality condition then stop. Otherwise each sub-problem is branched further.
6. Repeat this process of branching until all the sub-problems are explored or the stopping criteria is met.

2.3.2 Meta-heuristic Optimization Techniques

Heuristic means “to find”or “to discover by trial and error”. Heuristics are computational methods use to find good and feasible solutions to complex optimization problems, especially many real-world problems that are combinatorial in nature [116]. It has been proven from literature

that heuristics algorithms usually perform best for real-world problems given real-world time limitations. However, meta-heuristics are further development over the heuristic algorithms. Although, some use the two words interchangeably in literature. In recent literature, meta-heuristic is the general name given to all stochastic algorithms with randomness and local search. The word "meta-heuristic" was first introduced in [32], and was previously called modern heuristics [94]. The word "meta" before the heuristic simply means 'beyond' or 'higher level' which implies that meta-heuristics perform better than simple heuristics. Two essential components of meta-heuristics which determine their behaviours are intensification and diversification [13, 33, 130], but the balance between these two components is pivotal to the quality of any meta-heuristic solution. Figure 2.3 still gives an overview of the different meta-heuristic algorithms and their classifications. Examples of meta-heuristic algorithms are GA, Ant Colony Optimization(ACO), PS, Bee Algorithm(BA), Particle Swarm Optimization(PSO), Simulated Annealing(SA), Harmony Search(HS) and Firefly Algorithm(FA). The meta-heuristic algorithms used in this study are further explained in the next sections.

2.4 Genetic Algorithm

2.4.1 Introduction

GA was developed by John Holland and his partners in the 1960s and 1970s [27] [129]. GAs mimic the biological evolution of Charles Darwin's theory of natural selection. Holland was the first to use the genetic operators in his study of adaptive and artificial systems. The genetic operators which are selection, crossover and mutation form an important part of a GA to solve optimization problems. GA has these two noticeable advantages [129], aside many other advantages over traditional optimization algorithms, which are: the ability to tackle complex optimization problems and parallelism. GAs can be used to solve any type of optimization problems whether the objective function is linear or non-linear, changes with time or not, continuous or discontinuous.
However, an appropriate choice of the important parameters is essential for the algorithm to converge [129]. Some of these issues that can affect the performance of GA borders on the formulation of fitness function, the value of the population size, the rate of mutation and crossover, and the type of selection. This section gives a brief explanation of the various types of encoding structures that are often used in GA, the operators in GA, the pseudo-code for the GA and some applications of GA as presented in various literature.

2.4.2 GA Encoding

Encoding solutions of an optimization problem into chromosomes is a central issue in GA. GA cannot be implemented without a method to encode potential solutions of the problem to a form a computer can process. There are different encoding methods that have been created for certain

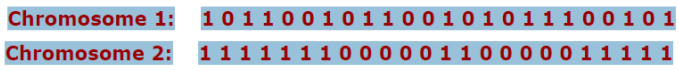

Figure 2.4: The chromosome of a binary encoding

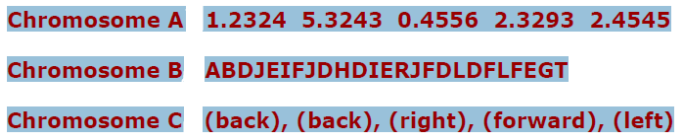

Figure 2.5: The chromosome of a value encoding

optimization problems for effective implementation of the GAs. The following are some of the classification of encoding methods used in GAs as presented in literature.

2.4.2.1 Binary Encoding

Binary encoding, a string of 0s and 1s which represent the value of each solution, is the most common type of encoding in use. This can be attributed to the fact that Holland and his students in their earliest work focused on such encoding or partly because many of the existing GA theories are centered on the premises of using binary encoding. However, binary encoding is not natural for some optimization problems. Hence, the need to sometimes make corrections after the genetic operation is completed [80]. An example of that is given in Figure 2.4

2.4.2.2 Value Encoding

Every chromosome in value encoding is a string of values. The values are connected to the problem at hand which makes the use of binary encoding for these type of problems difficult. Value encoding gives better results than binary encoding on some class of optimization problems such as function optimization and COPs. However, this type of encoding often requires a new genetic operator to be used. Figure 2.5 gives some examples of value encoding.

2.4.2.3 Permutation Encoding

Permutation encoding is best used for problems that search for the best permutation or combination of items subject to some constraints. This type of encoding is only useful for ordering problems such as the TSP and the eight queen problem. Since the order of items is important, it is necessary to use some type of crossover and mutation operator to keep the chromosome consistent or feasible. An example of permutation encoding is shown in Figure 2.6

Figure 2.6: The chromosome of a permutation encoding

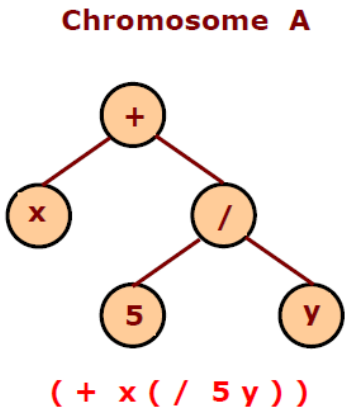

Figure 2.7: The chromosome of a tree encoding

2.4.2.4 Tree Encoding

Tree encoding represents every chromosome as a tree of some objects. This type of encoding is useful for evolving programs or structures that could be encoded in trees. Lisp and other functional programming languages are suitable for this encoding. See Figure 2.7 for an example of this.

2.4.3 GA Operator

Genetic operators are used to maintain genetic diversity which are essential for evolutionary processes in GAs. These genetic operators are similar in operation to the ones in nature. The following explains briefly the selection, crossover and mutation operators.

2.4.3.1 Selection

In selection, individuals are taken to be parents and they are recombined to produce children for the next generation. This selection is done to favour the chosen of fitter individuals. The fitter individuals have better solutions than the other ones. Better solutions mate more often than the poor solutions, if at all they mate. Hence, a mating pool is formed. Then offspring are produced using the crossover and mutation operators by the recombination of these solutions in the mating pool. One important issue in selection operators is the selection pressure. The selection pressure is the degree to which the fitter individuals are favoured. The higher the selection pressure, the more the fitter individuals are chosen. GAs improve the population fitness using this selection

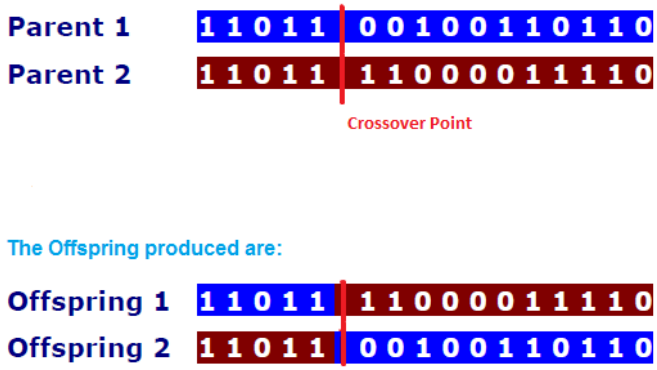

Figure 2.8: An example of a single point crossover

pressure over the subsequent generations. The magnitude of the selection pressure influences the convergence rate of GA, with higher selection pressures resulting in higher convergence rates [80]. On the other hand, the genetic diversity will be much higher if the selection pressure is too low, which will cause the algorithm to take a longer time to converge to an optimal solution. Hence, it is important to set the selection pressure appropriately.

The selection operator determines the manner in which individuals are chosen for the mating pool [41]. Many selection operators exist in literature [80]. Although, they vary in complexity, they are essentially doing the same thing. They pick from the current population in a random way the strings that have fitter individuals and introduce their multiples into the mating pool. The most commonly used selection methods are: *Stochastic Universal Sampling*, *Roulette Wheel Selection*, *Sigma Scaling*, *Elitism*, *Boltzmann Selection*, *Rank Selection*, and *Tournament Selection*.

2.4.3.2 Crossover

Two individuals that are selected at random from the mating pool are acted upon by the crossover operator to give some new offsprings. The crossover operators provide exploration of the search space by introducing some new individuals into the population. Similar to the evolution in nature, crossover uses the idea that new offspring may be better than their parents if they take the best features from each of the parents [14]. Crossover occurs according to pre-defined crossover probability. A simple example of a crossover technique is the single-point crossover, also called one point crossover. This crossover technique has the chromosome of each parent cut randomly at a defined position and the tail part of the chromosome are swapped to produce two new offsprings (see Figure 2.8). The simple crossover technique is usually effective but not as effective as other more complicated operators in some problems.

Crossover ensures that the genes of individuals from the mating pool are recombined to produce new solutions to the problem. Some genes or parts of genes from each parent are introduced into

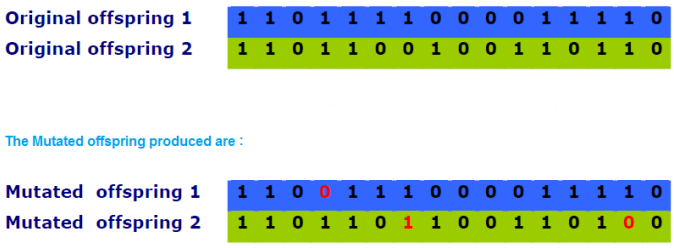

Figure 2.9: An example of a simple mutation

each of the offsprings. In order to maintain an evolutionary analogy that is competitive from one generation to another, the crossover operator is only applied to a mating couple with a probability *pc*, which is often a real number between 0.6 and 1.0 [41]. Other types of crossover operators aside the single-point crossover are: *Two Point Crossover*, *Uniform Crossover*, *Arithmetic Crossover*, and *Heuristic Crossover*.

2.4.3.3 Mutation

Mutation operator is usually applied after crossover to each offspring. Mutation keeps the algorithm from being trapped in a local optimum. It helps to recover lost genetic materials and to introduce new genetic structures in the population by randomly disturbing the genetic information [80]. It also ensures that there is a non-zero probability of obtaining a solution from the population. Mutation is used to maintain genetic diversity in the search space in order to ensure that all the points are reached [41]. A simple mutation operator has a gene changed with some small probability *pm* to each child for binary coding. This is called flip bit mutation and it is shown in Figure 2.9. The mutation operators are of different methods which are selected according to the way the chromosomes are encoded. Other types of mutation operators are *Boundary mutation*, *Non-Uniform mutation*, *Uniform mutation*, *Gaussian mutation*, and *Adaptive Feasible Mutation*.

2.4.4 Application of GA

GAs have been used in different fields due to its nature-inspired capability to find some promising regions in a large search space. Various areas of study where GAs have been applied and are still being applied include the following: In optimization, GAs have been used for multiple choice optimization problems [112], as a decision support tool to optimally allocate available resources [119], to search for near-optimum solution of the resource allocation and levelling simultaneously [108]; in robotics, GAs were used to address the problem of automatic parking by a back-wheel drive vehicle [19], for robot navigation controller (Neural-network based) optimization [98]; in point-to-point trajectory planning for a 3-link (redundant) robot arm [46]; in machine learning, GAs have been used for designing neural networks [104], for classification and prediction [107], to optimize the board evaluation function of the game of draughts (checkers) [51], for data optimization of

efficient results estimation alongside back propagation neural network algorithm [131]; in signal processing, GAs have been used for optimizing the coefficients of recursive digital filters [106], to choose appropriate values for the parameters of the modified varri signal segmentation method [44], for the improvement of the fidelity of the watermarking scheme [86], to address the problem in control engineering and signal processing [35]; in design, GAs have been used to generate the topological arrangement of spaces in an architectural layout [61], to determine a set of unknown parameters that best matches the Blaze II chemical laser model with experimental data [64], for the optimization of 66 setup parameters for a simulation of a formula one car [62], to develop neural reinforcement controller [77], to obtain near optimal solutions to the vehicle routing with time and capacity constraints [54]; in automatic programming, GAs have been used to search an extremely large solution space of all possible Module Dependency Graphs (MDGs)[18], to synthesize circuits to meet any specification [11]; in Economics, GAs have been used to model resource exploitation under bounded rationality [99], to find the most profitable trading rule and to calculate the most appropriate trade time [29, 4], to investigate economic evolution[100].

2.5 Pattern Search

2.5.1 Introduction

PS is a direct search or derivative-free optimization method for solving a given optimization problem. As a derivative-free algorithm, it works well on objective functions that are non-differentiable, stochastic, and discontinuous. It could also be used to find a global optimum of a function. PS has been proved to be efficient, less computationally expense, and to converge in its implementation to optimization problems [73, 113, 1, 47]. Hence, it is gaining more attention among researchers working on various optimization problems. PS algorithm searches a set of points known as patterns around the current point, looking for one where the value of the objective function is minimum than the value at the current point [73, 113, 1]. These set of points expand or shrink depending on the objective function value at the current point [79, 1, 78]. The search stops after a minimum pattern size is reached. As a heuristic method, PS provides good approximate solutions for many optimization problems although it could fail on others. Hooke and Jeeves coined the name pattern search in 1961 [91]. The PS method is often referred to as the Generalized Pattern Search(GPS) method in literature [3]. The following two definitions [16, 1] are essential in understanding the search directions of PS algorithm.

Definition 2.1. A positive combination of the set of vectors $D = \{d_i\}_{i=1}^{r}$ is a linear combination $\sum_{i=1}^{r} \lambda_i d_i$, where $\lambda_i \geq 0$, $i = 1, 2, ..., r$.

Definition 2.2. A finite set of vectors $D = \{d_i\}_{i=1}^{r}, n+1 \leq r \leq 2n$, forms a positive spanning set for $\Re^n$ if any $v \in \Re^n$ can be expressed as a positive combination of vectors in D. The set of

vectors D is said to positively span $\Re^n$. The set D is said to be a positive basis for $\Re^n$ if no proper subset of D spans $\Re^n$

With the understanding of these definitions, the search directions used by PS algorithm is described next. The simplest search direction is made up of $r = 2n$ vectors and it is given by

$$D = \{e_1, ..., e_n, -e_1, ..., -e_n\} \tag{2.24}$$

where e_i is the ith unit coordinate vector in $\Re^n$ and D is a set with a maximal positive spanning direction.

2.5.2 Application of PS

The PS optimization method could be applied to solve a diverse number of optimization problems that might not be solved by some standard optimization methods [96]. It is easy in concept and computationally inexpensive. A detailed review of direct search methods is given in [74] for unconstrained optimization. The following are few of the applications of PS algorithm to real world optimization problems. Alsumait et al. [96] applied the PS algorithm to solve power system Economic Load Dispatch problem (ELD) with a valve-point effect. The algorithm was used to minimize the power generation cost function, estimated as a quadratic function, subject to some given constraints. The results suggested that the PS algorithm could obtain a better optimal solution within a reasonable time than Evolutionary Programming(EP) and GA. Nicosia and Stracquadanio [40] also used the generalized PS algorithm in predicting the tertiary structure of small molecules, called peptides. The study explained that proteins are organized in a three-dimensional structure called the tertiary structure[40]. And that the understanding of this could make the treatment of diseases easier. This is one of the leading problems in structural bio-informatics. The peptide structure prediction was modeled as a nonlinear optimization problem which minimizes the potential energy function called the Ecepp/3 function. The results obtained were compared with the PEPstr algorithm, a state-of-the-art algorithm, and was found to be 21.17% better in terms of the average root mean-square deviation.

Furthermore, Song et al. [128] implemented the PS algorithm to enhance the performance of high-frequency surface wave dispersion curves which are highly nonlinear and multimodal. The study investigated the PS algorithms and found them to be better in their application to rayleigh wave inversion compared to GAs. It was suggested that the implementation of PS using the GPSPositiveBasis2N as the inversion strategy could improved the performance of the rayleigh wave dispersion curves [128]. Setting the expansion factor to 1 and contraction factor to 0.5 can improve calculation efficiency. Shabanzadeh et al. [88] investigated the problem of pattern classification in data mining which could be formulated as a non-smooth and non-convex optimization problem.

The unsupervised classification of patterns is known as clustering and is very useful in data analysis [88]. The study applied the PS algorithm to solve the partition clustering problem and was considered to give better results than the other algorithms.

2.6 Particle Swarm Pattern Search Algorithm

2.6.1 Particle Swarm Optimization

PSO is a population based stochastic optimization method developed by Eberhart and Kennedy [129] in 1995. PSO is inspired by the social behavior of bird flocking or fish schooling [126, 129]. It shares some similarities with GA, an evolutionary technique. However, PSO does not use evolution operators like most evolutionary computation techniques. The particles, which are the potential solutions of the PSO, search through the problem space by following the best current particle [126]. The algorithm starts by generating a population of candidate solutions, called the swarm. Each individual particle of the swarm moves within the search space, using both its knowledge and that of the swarm to search for a better solution and find an optimum value to the given problem. Hence, the knowledge of previous successful solutions is essential to the migration of the swarm. The position of each particle as well as the swarm is expressed in terms of its previous position, $x(t)$, at time t and the velocity, $v(t+1)$, of the particle which describes the direction in which they move. The position and velocity of a particle is defined as follows:

$$x(t+1) = x(t) + v(t+1) \tag{2.25}$$

$$v(t+1) = v(t) + c_1 r_1 (y(t) - x(t)) + c_2 r_2 (\hat{y}(t) - x(t)) \tag{2.26}$$

where c_1 and c_2 are positive real parameters called the cognition and the social parameters, r_1 and r_2 are random numbers on the interval (0,1) which ensure the exploration of the search space, $y(t)$ is the best previous position of that particle called the *pbest*, and $\hat{y}(t)$ is the best previous position of the swarm called the *gbest*. PSO has been widely used in literature due to the following reasons: it is faster than many other methods, it gives better results in a cheaper way, and it has few parameters to adjust. More details of PSO and other variants can be found in [129, 95, 127]

2.6.2 Particle Swarm Pattern Search

The Particle Swarm Pattern Search (PSwarm) algorithm, which was developed by Vaz and Vicente [47] in 2007, integrates the PSO into the PS framework that gives a better fitting for global optimization due to its high capability to explore the search space [47, 36]. The algorithm is a PS method which converges to stationary points from any arbitrary starting points with a particle swarm method applied in the search phase of the PS method for more aggressive global exploration of the objective function. It was shown from the numerical experiment performed in

[47, 36] that a larger part of the computational work is done in the search phase by the particle swarm. The resulting PSwarm algorithm still takes the basic structure of the PS algorithm which produces sequences of iterates along the conventional requirements for this class of method [47, 36].

Vaz et al. [36], broadened the PSwarm algorithm to handle general linear constraints. The poll step for the PSwarm contains positive generators for the tangent cone of the approximated active constraints, which also addresses the degenerate instance. An ellipsoid of maximum volume engraved to the feasible set is used to compute the initial population of the particle swarm in the search step. Each iterate of the PSwarm algorithm is divided into two steps namely: the search and the poll step. The coordinate search was applied in the poll step while the particle swarm was applied in the search step to generate points in the feasible region and equip the overall method to find a global minimizer. Particle swarm was used for this purpose due to the fact that it is a population-based method with fewer parameters to adjust and because it can be easily parallelized. The use of a population in the search step also helps in polling around the best particle in the poll step which enhances the robustness of the algorithm. The poll step reduces the number of particles by removing particles that are too close to each other in order to enhance their overall efficiency.

2.7 Penalty Method for Constrained Optimization

Constrained optimization problems are often converted to unconstrained ones to make it easier to solve such problems [8]. The CSPA problem is one of such constrained optimization problems. Hence, we shall discuss the penalty function method which is the oldest and more popular method for solving constrained optimization problems [15]. The penalty method converts the constrained optimization problems into equivalent unconstrained problems by adding to or subtracting from the objective function some positive terms called penalty terms whenever the constraints are violated, and adding or subtracting zero when the constraints are not violated [15, 70, 83]. This makes the solution of unconstrained problems to eventually converge to the solution of the constrained problems. The penalty functions were initially proposed by Richard Courant 1940s and were later extended by Caroll and Fiacco & McCormick. The rest of this section gives a detailed but brief explanation of different types of penalty function methods used for constrained optimization problems. Given a constrained minimization problem of this form

$$Maximize \qquad f(x) \tag{2.27}$$

subject to

$$g_i(x) \leq G_i \qquad where \;\; i = 1, 2, \ldots m \tag{2.28}$$

$$h_j(x) = H_j \qquad where \;\; j = 1, 2, \ldots n \tag{2.29}$$

$$x \geq 0 \tag{2.30}$$

A penalty function method can transform the constrained optimization problem into an unconstrained problem using the additive form or the multiplicative form.

1. The additive form

$$Fitness(x) = \begin{cases} f(x) & \text{if } x \text{ is feasible} \\ f(x) + \alpha_1 p(x) & \text{otherwise} \end{cases} \tag{2.31}$$

2. The multiplicative form

$$Fitness(x) = \begin{cases} f(x) & \text{if } x \text{ is feasible} \\ f(x) * \alpha_2 p(x) & \text{otherwise} \end{cases} \tag{2.32}$$

where $p(x)$ is the penalty function, and α_1, α_2 are penalty factors.
Although the additive form is the popular type used in literature, the multiplicative form can also converge to a solution for the constrained problems. There are different types of penalty methods as seen in literature. A brief explanation and mathematical implementation of these different types of penalty methods are presented as follows.

2.7.1 Death Penalty

The death penalty is a type of penalty method in which infeasible solutions are out-rightly rejected [15, 83]. It does not use any information from the infeasible solution of the problem, hence, if there are no feasible individuals at the first population, the process will not continue [115, 132, 135, 134]. The formulation could be written as

$$Fitness(x) = \begin{cases} f(x) & \text{if } x \text{ is feasible} \\ 0 & \text{otherwise} \end{cases} \tag{2.33}$$

This method can only work better when a reasonable portion of the search space or the whole search space is convex.

2.7.2 Static Penalty

Unlike the death penalty method, the static penalty method does not reject the infeasible solutions but uses a constant penalty factor which does not change with the iteration number throughout

the entire computational process to penalize infeasible solutions. The most popular of this function was proposed in [5] as follows

$$Fitness(x) = f(x) + \sum_{i=1}^{m}(R_{ij} * \max[0, g_i(x)]^2) \tag{2.34}$$

where R_{ij} represents the coefficient corresponding to the jth constraints and ith violation level, m is the number of constraints. Some other static penalty methods used in literature can be seen in [34, 24]. Although, the major disadvantages of the static penalty method is that high number of parameters need to be set to get a good feasible solution, this study uses the static penalty method to convert the constrained CPSA problem to an unconstrained one. The reasons for this are: (1) The static penalty method is simple and easy to implement, (2) It is one of the most popular types of penalty function methods used in literature and (3) It is believed that if the static penalty method works well with this problem, then other complex types of penalty function methods would also work.

2.7.3 Dynamic Penalty

This type of penalty method is dynamic due to the changing nature of the penalty factor over time. As the generation number of the algorithm increases, the penalty factor also increases, making it difficult for infeasible solutions to be present as the generation number increases. In [52], the dynamic penalty was formulated as follows

$$Fitness(x) = f(x) + (C * t)^{\alpha} * SVC(\beta, x)) \tag{2.35}$$

where

$$SVC(\beta, x) = \sum_{i=1}^{n} D_i^{\beta}(x) + \sum_{j=1}^{p} D_j(x) \tag{2.36}$$

and

$$|x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x \leq 0 \end{cases} \tag{2.37}$$

$$D_i(x) = \begin{cases} 0 & g_i(x) \leq 0, 1 \leq i \leq q \\ |g_i(x)| & \text{otherwise} \end{cases} \tag{2.38}$$

$$D_j(x) = \begin{cases} 0 & -\varepsilon \leq h_i(x) \leq \varepsilon, q+1 \leq j \leq m \\ |h_i(x)| & \text{otherwise} \end{cases} \tag{2.39}$$

The constants C, α and β are to be determined by the user. However, the values suggested to be used in [52] were C = 0.5, $\alpha = 1$ or 2 and $\beta = 1$ or 2.

The dynamic penalty method was also implemented in [102] using

$$Fitness(x) = f(x) + V(g)[A\sum_{i=1}^{m}(\delta_i W_i \phi(d_i(S))) + B]\delta_s \tag{2.40}$$

where

A = severity factor

m = total number of constraints

$$\delta_i = \begin{cases} 1 & \text{if constraint } i \text{ is violated} \\ 0 & \text{otherwise} \end{cases} \tag{2.41}$$

W_i = weight factor for constraint i

$d_i(S)$ = the degree of violation of constraints i introduced by solution S

$\phi(d_i(S))$ = function for the degree of violation of constraints i introduced by solution S

B = penalty threshold factor

$$\delta_s = \begin{cases} 1 & \text{if S is feasible} \\ 0 & \text{otherwise} \end{cases} \tag{2.42}$$

V(g) = an increasing function of the current generation g in the range(0,..,1) with the best performing function designed as V(g) = $(\frac{g}{G})^2$ and G is the total number of generations.

2.7.4 Annealing Penalty

The annealing penalty method was developed in [133] based on the idea of simulated annealing [103]. This method is also called GENECOP II, which means the second version of the GEnetic algorithm for Numerical Optimization for COnstrained Problems. The steps of this method as outlined in [132] are as follows:

- Dissever the constraints into four subsets: linear equations, linear inequalities, non-linear equations and non-linear inequalities
- Choose a random initial point that satisfies the linear constraints as the starting point. The initial population will consist of copies of this point
- Set $\tau = \tau_0$ which is the initial temperature
- The population is evolved using the formula

$$fitness(x, \tau) = f(x) + \frac{1}{2\tau}\sum_{j \epsilon A} f_j^2 \tag{2.43}$$

- if $\tau < \tau_f$ stop

- else
  - reduce τ
  - use the best solution as a starting point for the next generation
  - reiterate the previous steps

The values of the initial temperature, next temperature and the freezing point was set in [133] as $\tau_0 = 1$, $\tau_{i+1} = 0.1\ \tau_i$, $\tau_f = 0.000001$. This method deals with active constraints only at each iteration. The pressure on the feasible solutions increase as the temperature τ decreases.
Carlson [25] also suggested the evaluation of the fitness value using the formula

$$fitness(x) = A.f(x) \tag{2.44}$$

where

$$A = \exp^{\frac{-M}{T}} \tag{2.45}$$

and

$$T = \frac{1}{\sqrt{(t)}} \tag{2.46}$$

The annealing function, A, depends on M, which is the measure of the amount constraint violation and T, the cooling schedule which depends on the running time of the algorithm.

2.7.5 Adaptive Penalty

The adaptive penalty method was developed in [10]. The method updates the penalty parameters based on the feedback it gets from the search process. Each individual is evaluated by the following formula as suggested in [10]

$$fitness(x) = f(x) + \lambda(t)[\sum_{i=1}^{q} g_i^2(x) + \sum_{j=q+1}^{m} |h_j(x)|] \tag{2.47}$$

where
$\lambda(t)$ is obtained for each generation using

$$\lambda(t+1) = \begin{cases} \frac{1}{\beta_1}\lambda(t) & \text{if all the best individuals are feasible} \\ \beta_2\lambda(t) & \text{if all the best individuals are infeasible} \\ \lambda(t) & \text{otherwise} \end{cases} \tag{2.48}$$

This means that if all the best individuals of the last generation are feasible, the penalty factor $\lambda(t+1)$ for the next generation decreases. But if they are not feasible, the penalty factor $\lambda(t+1)$

increases. However, if the best individual consists of both feasible and infeasible solutions, then the penalty factor remains the same. It also has the conditions that $\beta_1, \beta_2 > 1$, $\beta_1 > \beta_2$ and $\beta_1 \neq \beta_2$ Other earlier works on adaptive penalty function method can be seen in [28] using the formula

$$fitness(x) = f(x) + (f_{all} - f_{feas}) \sum_{i=1}^{m} (\frac{\Delta b_i(x)}{\Delta b_i^{nef}})^k \tag{2.49}$$

where

$\Delta b_i(x)$ = value of violation for constraint i
Δb_i^{nef} = near feasibility threshold for constraint i
k = severity parameter
f_{all} = objective value of the best unpenalized solution found
f_{feas} = objective value of the best feasible solution found

Further work on this can be seen in [111] where the multiplicative form of the adaptive penalty method in [28] was considered and is given by the formula

$$fitness(x) = f(x) + (1 - \frac{1}{m} \sum_{i=1}^{m} (\frac{\Delta b_i(x)}{b_i(x)})) \tag{2.50}$$

where

$$\Delta b_i(x) = max\{0, g_i(x) - b_i\} \tag{2.51}$$

And in [70], the multiplicative form of this method was improved by giving high penalties for infeasible solutions.

2.8 Conclusion

This chapter started with an introduction to optimization; a simple definition of an optimization model which comprises of three essential components namely: the objective function, decision variables and a set of constraints to be satisfied; and a classification of the optimization problems as suggested in [8] was discussed. Following this, some common examples of COPs were briefly explained as presented in literature. Classifications of techniques explored by scientists and researchers in solving optimization problems were also discussed. While some of these techniques can obtain optimal solutions (such as the exact techniques) to a given problem, others can simply obtain a good solution, though not optimal, examples of such are the heuristic techniques. Finally, a constraint handling techniques commonly used for constrained optimization problems in literature were also examined. In the next chapter, the campus space allocation problem with respect to

the optimization model employed in addressing the problem as well as the solution steps developed for the problem are discussed.

Chapter 3

Campus Space Allocation Problem

> "In a competitive world, only the best (safest, cheapest, fastest, ...) is good enough ", [110].

3.1 Introduction

The problem of allocating parking spaces to users on campus is a difficult one especially when it is to be done for a large number of users. Like any other COPs, it is a NP-hard problem. It is difficult in the sense that the allocation must not be done in an indiscriminate way [7], but rather on a selective basis so as not to deny many users the privilege of having parking spaces close to their destinations [7]. Hence, an optimization model is employed for this problem since it ensures efficient use of parking facilities by determining the optimal allocation of parking spaces to users [118]. This chapter reviews similar study on campus parking and presents the optimization model for addressing the campus parking allocation problem. It also explains the solution ideas that will be used in addressing the problem and gives a description of the exact method as well as the pseudocode of the meta-heuristic algorithms employed for the solution of the problem.

3.2 Related Works on PSA

In the previous studies on parking problem, Goyal and Gomes [58] formulated a linear programming model for determining the optimum allocation of existing car parking facilities on campus for multiple classes of users. The model was a minimization model for the distance walked by different users from their allocated parking lot to their office destination. The paper explained that the previous models on campus parking spaces allocation assumed a single class of user which was not so in a real life situation. But the model proposed was for multi-class users in a closed community like the university environment. Three different cases for the model were considered.

The first case is when the number of users is greater than the number of available parking spaces on campus. The second case is when the number of parking spaces is greater than the number of users. And the third case is when the number of users equals the number of parking spaces on campus. While a user can be sure of getting a parking space on campus for both the second and third cases, the user is not sure of getting a place for parking in the first case. Hence, a suggestion on how to calculate the parking permit for the first case is given.

Batabyal et al. [2] discussed the probabilistic nature of parking demands and the violation of parking rules. The paper further explained that it is important to understand this probability nature of parking demands and parking rules violation due to the tripartite demand for parking by the faculty, staff and students which has created a limited supply of parking spaces in the university and is a major source of concern for parking planners and managers on how best to allocate these limited parking spaces to users in the university. The paper focused on two parking issues in the university environment namely: short term and long term parkers, and parking rules violators. For the short term and long term parkers, the mean parking time was determined and was used to calculate the probability distribution function of the number of occupied parking spaces at a given time. And for the parking rules violators, the probability distribution function of the number of violators who were given a fine at inspection for violating the parking rules was calculated. The study did not differentiate the academic staff from the non-academic staff. The study considered the length of the parking time as a measure of whether a user is a long term or short term parker.

Narragon et al. [7] developed a probability model that allows different classes of users to be considered at the same time so as to assess alternative policies for issuing a higher number of parking permits than the number of available spaces in the parking lots. The model was implemented with data from a case study. The paper explained that the model could also be applied to an inventory problem, staffing of maintenance facilities problem and staffing a stenographic pool problem. It was further expatiated that the policy of allocating one space per user in many institutions will only lead to more parking problems resulting in many vacant spaces that cannot be used by others. Hence, drivers are left with the option of parking in parking lots that are very far from their destinations (if at all they find one). The paper emphasized that there are only two major ways of addressing the demand for parking spaces on campuses; building more parking facilities or the efficient utilization of parking facilities. However, while the former appears to be a better solution to the problem, it is difficult to implement due to limited spaces and funds, this leaves the second option of ensuring that the existing facilities are effectively and efficiently utilized through policies that encourage such. Issuing more parking permits than the available parking spaces is called lot over-issuance or lot overselling [7]. The study emphasized that lot overselling has a potential of improving the use of parking spaces since it is based on the principle that all the spaces

in a parking lot are not totally occupied at any given time. Hence, it is possible to allocate more users to a parking lot without causing any parking conflict.

Mouskos et al. [17] formulated a deterministic dynamic Parking Reservation System (PRS) for performing parking space assignment on the minimization of parking costs in order to aid users in securing a parking space either before or during their trip. The mathematical model is formulated as an assignment binary integer linear program which can be solved by any standard linear programming software. It was noted that searching for parking spaces in urban areas often results in the loss of substantial amount of time, congestion, environmental pollution and frustration by drivers. Searching for parking spaces adds more vehicles on the road and this impedes the speed of the drivers because the drivers would continue driving round "in circles "until they get a convenient parking space. Hence, the need for the development of the PRS to address this problem. The paper also explained that the previous methods such as the use of Parking Guidance and Information (PGI) systems or the Advanced Parking Information(API) systems that were used in addressing this problem could only help the drivers identify the parking lot they desire and the route they could take to get to the parking space. However, the PRS helps the drivers secure parking spaces before they leave their current position to the parking lot or when they are on the way to the parking lot.

As a follow up work to [17], Sun et al. [117] described how to improve the performance of terminal parking facilities and park-and-ride facilities by developing an on-line parking reservation system (PRS). It was emphasized that an improvement in terminal parking facilities will significantly reduce vehicles' congestion on the road and the travel time of drivers. While an improvement in park-and-ride will reduce the drivers' inconvenience, ease parking payment and reduce travel time. In implementing the web based system, the users were allowed to enter their information into the system, and then the system solved the parking reservation problem. After that, a shortest path problem was also solved using the Dijkstra's shortest path algorithm to determine the shortest walking path between the users' allocated parking lots and their destinations. A C language based ILOG CPLEX 9 optimizer integrated into a Java-based system using Extensible Markup Language (XML) files called Web-PRS, was used to solve the model. The Web-PRS is used to help both the subscribers (registered users) and the non-subscribers (non registered users) to make parking reservations on-line.

Brown-West [37] discussed the difficulties of finding a parking space near activity centres especially on campuses that are close to central business districts. An optimization model that considers some necessary parameters needed to obtain optimal parking by the traffic engineers and campus planners was developed and was used to determine the usage of existing land in a policy-

driven campus environment. The major objective of the optimization model was to determine the optimal arrangement of stalls that will maximize the number of parking spaces which will give minimum interference and travel discomfort to every user of the parking space. The paper explained that an increase in the size of a campus tends to increase the problem. Furthermore, other institutional needs on the campus were often given more attention than the need for parking spaces [37]. The study explained what the parking problem in the campus environment means, presented a generic optimization method that could help in managing parking spaces, introduced a multi-factor optimization model and created an awareness of the usefulness of the model to campus planners and traffic engineers. It was further expatiated that factors such as internal factors, external factors, campus size and growth, land availability, financial consideration, inappropriate design, and parking supply were the major influence on parking problems on campus. But some of these factors were often overlooked by the hired engineers when formulating the campus master plan and other planning documents.

Gracia et al. [90] applied an intelligent organizer of objects using a GA and a Hill Climbing(HC) algorithm to address the space allocation problem in a warehouse, a car parking and a land for cultivation. The study explained that the space allocation problem in a warehouse environment has a discontinuous available space because of some obstacles such as walls in the search space. In a car parking environment, it has a discrete available space. And in a land for cultivation environment, there is no restriction on the available space. However, the proposed intelligent system takes into consideration these restrictions in each of the three environments and it gave positive results when implemented.

Wang and Zhou [136] investigated some features of parking behaviour in Chang'an University campus which is similar to other chinese universities. They analysed some of the parking problems and recommended strategies for improving parking planning and management. The paper emphasized that the goal of managing parking facilities was to ensure that academic activities were carried out conveniently, since all the activities on campus were towards teaching and research. Some of the problems [136] identified were: limited parking places compared to parking demand, impediment of traffic due to the number of roadside parking, poor management of parking and gross parking facilities. Finally, the study suggested that the parking situation on campus could be improved by including parking planning into the campus planning, rationally distributing parking spaces using the principles in economics to control parking in order to find a good balance between demand and supply of parking spaces, and also by regulating parking behaviour to ensure users abide by the parking rules on campus.

Young et al. [118] presented a review of parking models to help in the understanding of the

interaction between traffic systems and parking systems. The parking models were grouped into three - choice, allocation and interaction models, in accordance with their objectives. The study discussed the connection between these three objectives leading to hierarchical models for parking analysis which help in the use of each model to address a particular policy situation. The first group in the parking model is the parking choice model which is further classified as the mode choice model and the location choice model. The study explained that drivers react to changes in parking policies and the availability or non-availability of parking spaces by finding alternative parking locations, changing the time of their trips, abandoning the trip entirely or changing their parking destinations [118]. It was further explained that the parking choice models help to model the behaviour of individuals or groups of people, this model makes use of available data, reduces data collection problems and transfers the observation made from one location to another. However, these models do not include the supply constraints which makes them limited to local optimal conditions, since they only investigate rates of parking demand within a particular parking supply.

The second group is the parking allocation model which is classified as the optimization model, constraint model, gravity model and traffic assignment model. The allocation models are concerned with allocating a given number of drivers to parking spaces. Among these parking allocation models, it should be noted that it is the optimization model that ensures the efficient use of parking facilities and that determines the optimal allocation of parking spaces or parking lots. The parking allocation models use supply constraints to direct traffic flows to the available parking spaces. This makes parking allocation models suitable for satisfying many parking policy requirements [118]. The third type of parking model is the parking interaction model [118]. The parking interaction models addresses the responses of drivers to parking policies. Two essential models that describe the parking interaction models are CLAMP (Computer-based Local Area Model of Parking behaviour) and ORIENT.

Sattayhatewa el al. [87] considered the use of three major factors, which are driving time, parking cost, and walking time, to evaluate the choice of parking lot in special occasions and analysed the consequence of such choices on the costs to everyone on the transportation network. The logit function is used in the formulation of the model for the parking lot choice while the user equilibrium and entropy maximization for trip distribution is used in the formulation of the joint parking lot choice and assignment model. The model was used to analyse the existing traffic conditions in such special events and to improve on the traffic conditions by efficient assignment of parking spaces for special events. The men's basketball at kohl centre in Madison, Wisconsin was used as the case study for the application of this model. The paper emphasized that in special occasions or events, the attendees of such events often seek to park their vehicles as close as possible to the location where the event is taking place. This often leads to overcrowding of the parking lots

close to such event's location while other parking lots, farther away, are not completely occupied. Many attendees spend valuable amount of time looking parking spaces. The results suggest that although the three factors are significantly important in parking lot choice, walking time is the dominant factor for initial preference for a parking lot.

3.3 Campus Parking Space Allocation Model

The model to allocate parking spaces on campus taking into consideration the reserved policy in a university environment is formulated as a linear programming problem. The model is an instance of the mathematical formulation given in [58] but with two classes of users: reserved users and unreserved users. A constraint to cater for the reserved allocation was introduced to the model in [58] where the number of users, T_U, is greater than the available parking spaces, T_S. The model is formulated as a linear programming model and it is given as:

$$Minimize \quad Z = \sum_{k=1}^{n} \sum_{j=1}^{m} \sum_{i=1}^{l} D_{jk} X_{ijk} \tag{3.1}$$

subject to:

$$\sum_{k=1}^{n} X_{ijk} = P_{ij} \quad \text{for } i = 1, 2, ..., l \text{ and } j = 1, 2, ..., m \tag{3.2}$$

$$\sum_{j=1}^{m} \sum_{i=1}^{l} X_{ijk} = A_k \quad \text{for } k = 1, 2, ..., n \tag{3.3}$$

$$\sum_{j=1}^{m} X_{ijk} \geq M_{ik} \quad \text{for } i = 1, 2, ..., l \text{ and } k = 1, 2, ..., n \tag{3.4}$$

$$X_{ijk} \geq 0 \qquad \forall \quad i, j, k \geq 0 \tag{3.5}$$

and

$$X_{ijk} \in Z+ \tag{3.6}$$

Where,

l = the total number of permit types (with index i)

m = the total number of users' buildings (with index j)

n = the total number of parking lots (with index k)

$$T_S = \sum_{k=1}^{n} N_k = \sum_{k=1}^{n} \sum_{i=1}^{l} M_{ik} \tag{3.7}$$

T_S = the total number of available parking spaces

N_k = the number of available spaces in the kth parking lot excluding the spaces for handicapped users

M_{ik} = the number of parking places available with permit type i in kth parking lot

A_k = the number of permits issued to the kth parking lot

D_{jk} = the distance between the jth building and the kth parking lot

X_{ijk} = the number of people having permit type i, working in building j and allocated to the kth parking lot

$$T_U = \sum_{i=1}^{l} B_i = \sum_{j=1}^{m} \sum_{i=1}^{l} P_{ij} \tag{3.8}$$

T_U = the total number of users demanding parking

B_i = the number of permit type i users

P_{ij} = the number of permit type i users working in building j

The objective function in Equation (3.1) minimizes the total distance walked by all the users from each parking lot to their respective buildings. Equation (3.2) is the permit type users constraint which ensures that the sum of the parking allocation in each parking lot is equal to the number of users with the permit type for the parking lot. Since several parking permits are issued for different parking lots, the constraint in Equation (3.3) ensures that the sum of parking allocation for users with permit type i working in building j is equal to the parking permit issued for the parking lot. The optimization model in Equation (3.1) - (3.3) is formulated in [58] but our model uses an added constraint which is Equation (3.4) for allocating reserved parking spaces in the university environment. Equation (3.4) is called the reserved spaces constraint. The constraint ensures that the sum of the parking allocation, X_{ijk}, for all users is equal to the number of available reserved spaces, M_{ik}, for reserved and greater than the number of available unreserved spaces. The non-negativity and integer constraint in Equation (3.5 and 3.6) keeps the variables equal to or greater than zero and to be integers.

With the following assumptions:

1. The shortest walking distance between each parking lot and the user's destination is known and it is taken by all users [58];

2. A user will either have a reserved parking permit type or unreserved parking permit type (that is $l = 2$).

3. The parking tariff for a similar parking permit type is the same whether the users are allocated to a parking space closer to their building or not;

4. The probability of a user bringing his car on a particular day and the probability of the user finding a space on that day is the same for all users [57, 58].

5. The criterion for determining the optimal solution is the minimization of the total distance walked by all the users [57].

Constrained CPSA Model into Unconstrained Model

The CPSA constrained problem must be transformed into an unconstrained optimization problem in order to conveniently obtain solutions to the problem using the meta-heuristics algorithms. The static penalty method is used, as explained in section 2.7.2, to transform the constrained CPSA model to unconstrained CPSA model because it does not totally discard an infeasible solution and it is easy to implement. A penalty is incurred if the solution violates any of the constraints. This, however, moves the infeasible solutions to feasible ones (as noted in section 2.7.2). The mathematical model in section 3.3 is transformed into an unconstrained optimization as follows:

$$Minimize \quad Z = \sum_{k=1}^{n}\sum_{j=1}^{m}\sum_{i=1}^{2} D_{jk}X_{ijk} + \alpha_t C_1 + \beta_t C_2 + \gamma_t C_3 \tag{3.9}$$

where:

$$C_1 = \sum_{t=1}^{2*m} \alpha_t * \max(0, |\sum_{k=1}^{n} X_{ijk} - P_{ij}|)^2$$

$$\text{for } \; i = 1, 2 \;\; , \;\; j = 1, 2, ..., m \;\; and \;\; t = 1, 2, ..., 2 * m \tag{3.10}$$

$$C_2 = \sum_{t=1}^{n} \beta_t * \max(0, |\sum_{j=1}^{m}\sum_{i=1}^{2} X_{ijk} - A_k|)^2$$

$$\text{for } \; k = 1, 2, ..., n \;\; and \;\; t = 1, 2, ..., n \tag{3.11}$$

$$C_1 = \sum_{t=1}^{2*n} \gamma_t * \max(0, (M_{ik} - \sum_{j=1}^{m} X_{ijk}))^2$$

$$\text{for } \; i = 1, 2 \;\; , \;\; k = 1, 2, ..., n \;\; and \;\; t = 1, 2, ..., 2 * n \tag{3.12}$$

$\alpha_t, \beta_t, \gamma_t$ are known penalty factors and are used when the constraints are violated. Different values for these penalty factors were experimented with to ensure feasible results and a penalty value of 200 was finally used for each of the three penalty factors. Equation (3.9) becomes the fitness function of the CPSA problem which will be implemented later in the study using some meta-heuristic algorithms in MATLAB programming language.

3.4 Solution Idea

The solution to the CPSA problem consists of four different stages, as shown in Figure 3.1. These stages are followed sequentially with stage 1 being the users application information stage. In this stage, applications for parking spaces are received from different users and the applications are divided into reserved or unreserved categories [30]. The users are asked to provide details of their rank and time spent in the university as part of the information in the application. Stage 2 is where the users weights are computed and data is sorted for further use. Often times, the number of users demanding for reserved parking are more than the number of available reserved spaces. Hence, the number of users to be considered for reserved parking are selected using the computed weight values. The other users who demanded for reserved parking but were not selected will be added to the number of users to be allocated unreserved parking while still keeping them on the waiting list [30]. Stage 3 is the development of the optimization model for the parking space allocation problem and the determining of the appropriate methods to use in addressing the problem are carried out at this stage. Lastly, Stage 4 collates the sorted data in stage 2 alongside other necessary data, implements the data using the optimization model and methods shown in stage 3 and yields output of the solution and allocation for each user.

However, it is difficult to illustrate stages 1 and 2 for large number of users especially when the number of users are greater than fifty (50). Hence stages 1 and 2 are carried out on a computer system. This study only focuses on stages 3 and 4, given that stages 1 and 2 have already been carried out on a computer system to save time and space of listing all the users greater than 50 for each dataset that will be considered in this study. Stage 3 is the focus of this Chapter while stage 4 is the focus of Chapter 4. Highlights of these four stages are given below as illustrated in Figure 3.1.

Stage 1

Stage 1 consists of the following items as noted in [30]

- Consider the N number of users' applications
- Divide the N users into categories - reserved and unreserved
- Consider the rank r_i of each user according to their categories
- Consider the time t_i each user had spent in the system

Stage 2

The following items are steps in stage 2 some of which are noted in [30]

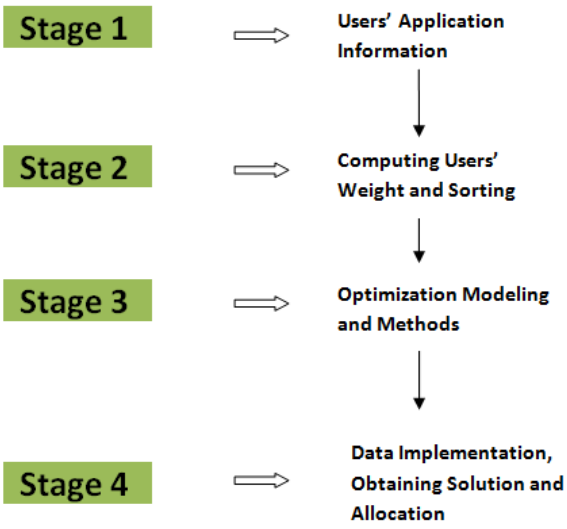

Figure 3.1: An overview of the solution stages of CPSA

- Calculate the weight w_i of each user using

$$w_i = r_i * t_i$$

- Determine the total number of available parking spaces for reserved and unreserved parking
- Select M (where M is the total number of available reserved parking spaces) users with the highest values of weights from the reserved category
- The remaining users that are not selected from the reserved category are added to the number of users for the unreserved category while still being kept on the waiting list.

Stage 3

- Developing an optimization model for the parking space allocation problem
- Determining the optimization method(s) to be used in solving the parking space allocation model

Stage 4

This stage is the data implementation stage and is where the model is actually solved, it consists of:

- Collating the complete data which consists of:
  - The available number of parking spaces in each parking lot
  - The distance D_{jk} from the office building B_j to the parking lots N_k for all users

  - The data sorted in Stage 2, which shows the number of users selected for reserved parking based on their weight and the number of users to be allocated unreserved parking (including the users being kept on the waiting list)
  - The number of parking permits to be issued for each parking lot

- Solving the CPSA model with the optimization method(s) decided in stage 3
- Allocation of users in each building to parking spaces. Users are allocated to the nearest parking lot according to their weights, w_i. The users with the higher weights are assigned to the nearer parking lots than those with lower weights.

3.5 Exact Optimization Solver

CPLEX solver is one of the fastest and most advanced optimization solvers in the world. It is a high performance software package that can be used to solve Linear Programming (LP), Mixed Integer Programming (MIP) and quadratic programming problems. The CPLEX solver can solve optimization problems ranging from hundreds to thousands to millions of variables. It was developed by Robert E. Bixby and was available commercially in 1988 by CPLEX Optimization Inc., which was acquired by ILOG in 1997; and was later taken over by IBM in January, 2009 [121]. The IBM ILOG CPLEX optimization solver (simply called CPLEX version 12.4) uses a variant of simplex method or the barrier interior point method to solve different kinds of optimization problems, see the IBM ILOG CPLEX website [121, 48] for more details. The CPLEX solver can either be used as a stand-alone interactive optimizer or can be called from a modeling layer that provides interfaces to the C++, C♯, Java, MATLAB and Python languages. It can also be accessed through some other independent modeling systems such as AIMMS, AMPL, GAMS, MPL, OpenOpt, OptimJ and TOMLAB.

AIMMS (Advanced Interactive Multidimensional Modeling System) is a modelling language [9] with a sophisticated Integrated Development Environment (IDE) and a well-developed user-friendly interface that fulfils the following criteria:

- Availability for MS Windows.
- A callable library or Application Programming Interface (API) that adds connectivity with other languages/environments.
- Reading or saving data from or to spreadsheets.
- Direct link to CPLEX and Gurobi solvers.
- Special ordered sets type 2 that make formulating piecewise linear functions easier.

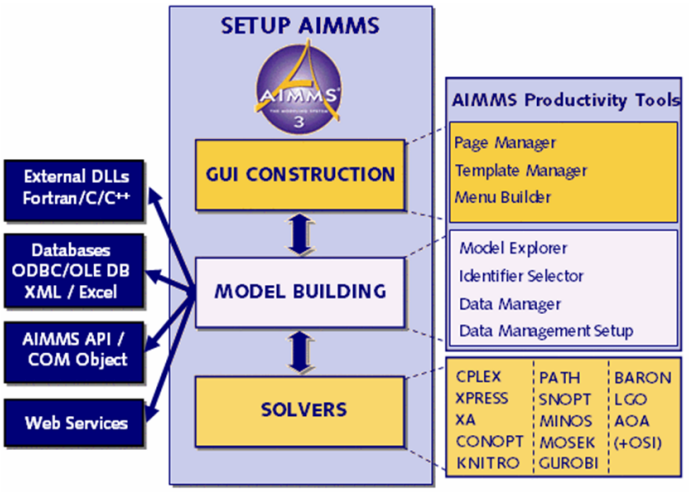

Figure 3.2: The components of AIMMS software

- Branching priorities of binary variables that can be manually set.

Figure 3.2 presents an overview of the components of AIMMS software. The CPLEX solver incorporated into AIMMS was used to obtain solution for each dataset.

3.6 Genetic Algorithm

GA is one of the population-based stochastic algorithms which generates near optimal solutions to optimization problems specifically to COPs using nature-inspired techniques such as inheritance, mutation, selection and crossover [112, 71, 27]. It belongs to the class of Evolutionary Algorithms (EA) [119, 27, 53]. GAs work on a population of individuals and consists of a string of genes called chromosomes to represent the candidate solution to the problem at hand. The genes could be a bit, an integer number, a real value or an alphabet character depending on the problem. GAs operators are then applied on these individuals to get new individuals. An important principle for developing and implementing GA for peculiar real word optimization problems is to find a good balance between exploration and exploitation of the search space. The Algorithm 1 represents the pseudo code of a GA [129].

GA generates a population of individuals, say $P(t)$, for generation t. Each individual is a possible solution to the given optimization problem. Each individual is evaluated to determine how fit it is to survive into the next generation. Two major types of transformations are performed on some selected individuals, they are: crossover, which creates new individuals by combining the genes from at least two individuals; and mutation, which creates new individuals by mutating some genes in a single individual. The new individuals that are created are called offspring $C(t)$ which

Algorithm 1 Genetic Algorithm

```
Objective function f(x), x = (x_1, ..., x_n)^T
Encode the solution into chromosomes (binary strings)
Define fitness F
Generate the initial population
Initialize the probabilities of crossover (p_c) and mutation (p_m)
while t < Max number of generations do
  new solution by crossover and mutation
  if p_c >rand then
    Crossover
  end if
  if p_m >rand then
    Mutate
  end if
  Accept the new solutions if their fitness increase
  Select the current best for new generation (elitism)
end while
Decode the results and visualization
```

are also evaluated. Then, a new and better population is formed by selecting fitter individuals from the parent population and offspring population. This process continues until after several generations or a stopping criteria is specified, to obtain the best individual that converges to an optimal or suboptimal solution of the problem. The flowchart in Figure 3.3 gives an overview of the GA structure. We implemented GA in MATLAB programming language making use of the capability of the MATLAB global optimization toolbox version 7.10 [78] to minimize the fitness function defined in Equation (3.9).

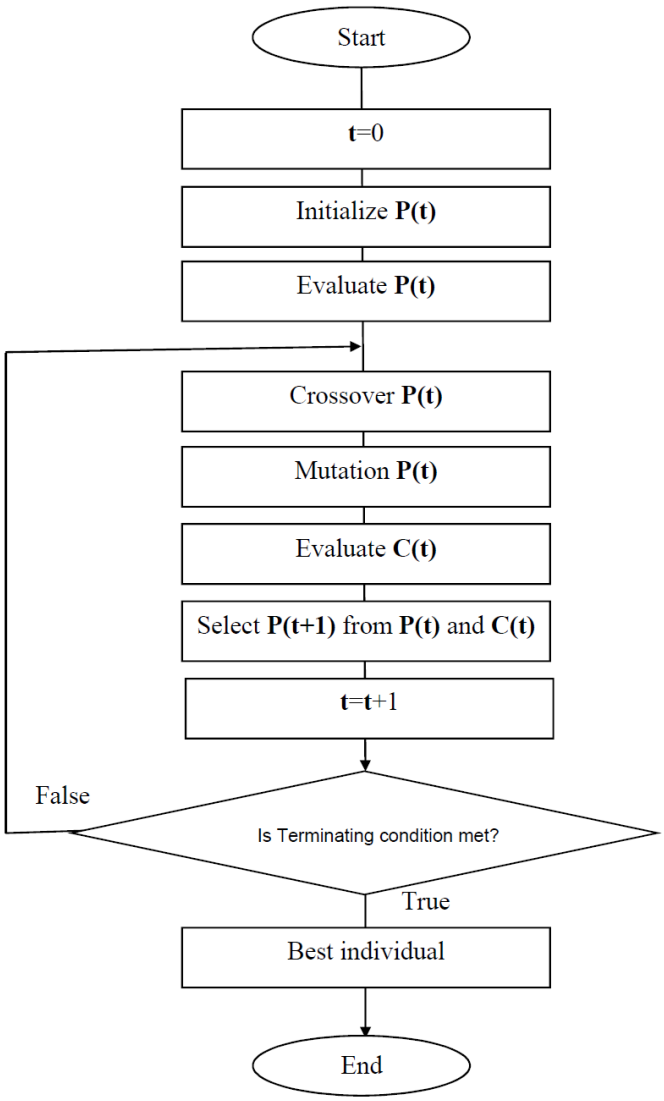

Figure 3.3: A flowchart explaining the structure of a genetic algorithm

3.7 Pattern Search Algorithm

This section presents an explanation of the PS algorithm. The algorithm generates a sequence of iterates $\{x_1, x_2, ...x_k, ...\}$ with non-increasing objective function values. There are two essential steps of the PS algorithm for each iteration k, which are the SEARCH step and the POLL step. Taking the value $r = 2n$ as earlier mentioned. The SEARCH step, which is the first step, evaluates the objective function at a finite number of points (say a maximum of V points) on a mesh (a discrete subset of $\Re^n$) in order to improve the current iterate. The mesh at the current iterate, x_k, is given by

$$M_k = \{y \in \Re^n | y = x_k + \triangle_k D_q : q \in Z_+^r\} \tag{3.13}$$

where y is a mesh trial point, $\triangle_k > 0$ is a step size control or mesh size parameter which depends on the iteration k, and Z_+ is the set of non-negative integers.
Generating trial points of the SEARCH step has no specific rules. Any strategy (including none) could be used to find a new point with a better function value than the current point [1]. The ultimate goal of the SEARCH step is to find a feasible trial point (on a mesh M_k) that gives a minima objective function value than the function value at x_k. The SEARCH step is tagged successful if there exists a feasible trial point $y \in M_k$, where y is one of the V points, such that $f(y) < f(x_k)$. Hence, y becomes the new iterate and the step size $\triangle_k$ is increased. Then the next trial point is chosen on a magnified mesh than the previous mesh. If the SEARCH step is unsuccessful, the second step, called the POLL step, is implemented before going out of the current iteration with the aim of decreasing the objective function value.

The POLL step is executed when the SEARCH step is unsuccessful. The POLL step searches the function about the current iterate x_k to generate trial points that give a new and better iterate. This produces a poll set, P_k. The poll set is made up of trial points that are placed at a step $\triangle_k$ away from the current iterate x_k, in the direction specified by the columns of D. The poll set, P_k, can be defined as:

$$P_k = \{p_i \in \Re^n | p_i = x_k + \triangle_k d_i \in D, i := 1, ..., r\} \tag{3.14}$$

where p_i is a trial point in the POLL step. The order taken by the points in P_k could be different and it makes no changes on its convergence. The pseudo code for the PS Algorithm [85] with both the SEARCH and POLL step is presented in Algorithm 2 below. We implemented the PS algorithm in MATLAB programming language making use of the capability of the MATLAB global optimization toolbox version 7.10 [78] to minimize the fitness function defined in Equation (3.9).

Algorithm 2 Pattern Search

Step 1. **Initialization**

Given x_0 (initial point in $\Re^n$), $\triangle_0 > 0$ (initial step size)

Where D is a finite set of positive spanning directions, x_k is the current iterate

Z is the set of non-negative integers

while Max number of generations not reached **do**

Step 2. **SEARCH step**

Evaluate f at points in the mesh M_k as defined by (3.13)

Search $y \in M_k$ such that $f(y) < f(x_k)$

if $f(y) < f(x_k)$ **then**

Set $x_{k+1} = y$

Mesh expansion: Let $\triangle_{k+1} = \theta_k \triangle_k$ (with $\theta_k > 1$)

Increase $k = k + 1$

The SEARCH step is **successful**, restart the search from this improved point

else

The SEARCH step is **unsuccessful**

Go to POLL step

end if

Step 3. **POLL Step**

Evaluate f at points in the poll set P_k as defined by (3.14)

if $f(p_i) < f(x_k)$ **then**

Set $x_{k+1} = p_i$

Mesh expansion: Let $\triangle_{k+1} = \theta_k \triangle_k$ (with $\theta_k > 1$)

Increase $k = k + 1$

The POLL step is **successful**, go to step 2 for a new iteration

else

Set $x_{k+1} = x_k$

Mesh reduction: Let $\triangle_{k+1} = \phi_k \triangle_k$ (with $0 < \phi_k < 1$)

Increase $k = k + 1$

The POLL step is **unsuccessful**, go to step 2 for a new iteration

end if

end while

3.8 Particle Swarm Pattern Search Algorithm

The PSwarm algorithm starts with an initial population and then applies one step of the particle swarm at each search step. If the search step succeeds, then more iterations of the particle swarm is performed so as to identify a neighbourhood for global minima. However, if the search step fails, the poll step is used to perform a local search at this point to the best position over all particles. The stopping criterion of the algorithm is the stopping criteria for both the particle swarm and PS methods. To obtain solutions with some degree of precision in PSwarm, particles that do not push the search towards global minima are removed. Particles are removed from the swarm whenever they are close to the $\hat{y}$, that is, when $\|y^i - \hat{y}(t)\| \leq \alpha(0)$. This means that they cannot further explore the search to a global optimum. Algorithm 3 gives the pseudo code of the PSwarm algorithm as implemented by [47, 36]

The hybrid algorithm combines the ability of particle swarm algorithm for global optimization with the PS rigorous method for local optimization which fosters global convergence from any starting point. The results obtained from the PSwarm solvers were compared with other global optimization solvers and found to be better in efficiency and robustness [47, 36]. This study also implemented the PSwarm solver, written in MATLAB m-files, as given by [36] to minimize the fitness function defined in Equation (3.9).

Algorithm 3 PSwarm algorithm

Choose the stopping tolerances $\alpha_{tol} > 0$ and $v_{tol} > 0$

Choose the initial population size s , set $I = 1, ..., s$

Calculate (randomly)the initial feasible swarm position $x^1(0), ..., x^s(0)$ and the initial swarm velocities $v^1(0), ..., v^s(0)$

Set $y^i(0) = x^i(0)$, $i = 1, ..., s$ and $\hat{y}(0) \in argmin_{z \in \{y^1(0), ..., y^s(0)\}} f(z)$

Let t = 0

SEARCH Step

Set $\hat{y}(t+1) = \hat{y}(t)$

For all $i \in I$ (for all particles) do:

if $f(x^i(t)) < f(y^i(t))$ **then**

Set $y^i(t+1) = x^i(t)$ (update the particle i best position)

if $f(y^i(t+1)) < f(\hat{y}(t+1))$ **then**

Set $\hat{y}(t+1) = y^i(t+1)$ (update the particle best position, search space and iteration) successful

Set $\alpha(t+1) = \phi(t)\alpha(t)$ (optionally expand the mesh size parameter)

end if

else

Set $y^i(t+1) = y^i(t)$

end if

POLL Step

if there exist $d(t) \in D$ such that $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) < \hat{f}(\hat{y}(t))$ **then**

Set $\hat{y}(t+1) = \hat{y}(t) + \alpha(t)d(t)$ (update the leader particle position; poll step and iteration successful)

Set $\alpha(t+1) = \phi(t)\alpha(t)$ (optionally expand the mesh size parameter)

else

Set $\hat{y}(t+1) = \hat{y}(t)$ (no change in the leader particle position; poll step and iteration unsuccessful)

Set $\alpha(t+1) = \theta(t)\alpha(t)$ (contract the mesh size parameter)

end if

Compute $v^i(t+1)$, and $x^i(t+1)$, $i \in I$

if $\alpha(t+1) < \alpha_{tol}$ and $\|v^i(t+1)\| < v_{tol}$ **then**

Stop

else

Increment t by one

Go to the SEARCH Step

end if

3.9 Data Representation for the CPSA Model

The data representation for the CPSA problem are based on the number of parking lots and the number of buildings in the campus location under consideration. A real value encoding is used for each of the variables. For example, if there are n parking lots and m buildings in a particular campus following the assumption that users can only demand for either reserved or unreserved parking spaces (that is, parking permit type is either reserved or unreserved, $l = 2$), then the total variable for such a case will be given as:

$$n * m * 2$$

Taking $n = 3$, $m = 5$, then the total number of variables for this instance will be 30. The chromosome representation for this instance is given in Figure 3.4. Where each row represents individual solutions to the CPSA problem.

This data structure shown in Figure 3.4 is used for all three algorithms. The operations on the data structure depend on the operation and procedure of each of the algorithms. For GA, the crossover operator and mutation operator are performed on the data structure. While for PSwarm algorithm, the particle swarm is applied on the data structure in the search step which will also be polled upon by the poll step, if the search step fails. However, for PS algorithm, the population of solutions is not permitted because PS evaluates a single solution at a time. Hence, the PS algorithm, unlike the other two, only takes an initial individual solution to work with.

The fitness value of each individual solution in the data representation is computed using the optimization model explained earlier and this is evaluated to determine how good each solution is. This structure of the fitness value is shown in Figure 3.5. This is only for GA and PSwarm algorithm. PS algorithm evaluates the fitness value of the initial solution with another solution obtained, the solution which minimizes the distance better is taken for comparison with the next solution.

$$\begin{bmatrix} var_{1,1}, & var_{1,2}, & var_{1,3}, \dots, & var_{1,30} \\ var_{2,1}, & var_{2,2}, & var_{2,3}, \dots, & var_{2,30} \\ var_{3,1}, & var_{3,2}, & var_{3,3}, \dots, & var_{3,30} \\ var_{4,1}, & var_{4,2}, & var_{4,3}, \dots, & var_{4,30} \\ var_{5,1}, & var_{5,2}, & var_{5,3}, \dots, & var_{5,30} \\ & & \dots & \\ & & \dots & \\ & & \dots & \\ var_{popsize,1}, & var_{popsize,2}, & var_{popsize,3}, \dots, & var_{popsize,30} \end{bmatrix}$$

Figure 3.4: The chromosome representation of the variables

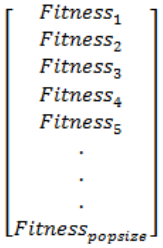

Figure 3.5: The fitness values representation

3.10 Conclusion

Previous related researches on campus space allocation problem have not implemented the heuristic algorithms for this kind of problem. Review of some of these literature were given at the beginning of the chapter. Then, the optimization model with an improvement in the constraints, to cater for the allocation of reserved spaces to demanded reserved users on campus, was presented. An overview of the stages involved in the approach to the CPSA was introduced, with the first stage being the receipt of parking space allocation requests from users in order of preference. Following this, the number of users to be given reserved parking allocations are selected using their weights based on the number of available reserved spaces in the university. Others that are not selected for reserved parking are added to those who initially demanded for unreserved spaces. The optimization model will then be implemented using the sorted data. The exact optimization solver - CPLEX incorporated in AIMMS sofware and the pseudo-code for the meta-heuristic algorithms - GA, PS and PSwarm, that will be used to address the CPSA problem were presented and explained in this chapter. The meta-heuristic algorithms do not require the derivative of the problem and PSwarm is the only hybrid algorithm among the three algorithms. The next chapter gives the experimental settings and the results of the CPSA problem using the CPLEX solver and the outlined meta-heuristic algorithms.

Chapter 4

Experimental Setting and Results

> "In particular, if algorithm A outperforms algorithm B on some cost functions, then loosely speaking there must exist exactly as many other functions where B outperforms A ", [45].

4.1 Dataset

Although the data in [57] did not include reserved parking, the dataset in this study follows some of its pattern. The data generated for this study are random variants of real life datasets collected from the risk management services of the University of Kwazulu-Natal [84] which take into consideration the reserved policy on campus. The motivation for using a random variant of a real life dataset is that: (1) They are good for research-oriented experiments because instances of every size and type can be generated, (2) They still preserve the main structure of the real life data with random changes in details, and (3) The real datasets are time consuming to collect. The University of Kwazulu-Natal made a new policy for staff parking fees and allocation of reserved spaces in 2006, which states that the percentage of open spaces to be considered reserved status would be 35.06% [84]. This parking reserved policy was put into consideration when writing the code for the generated datasets in all cases.

There are four different datasets used for this experiment. Each dataset has the distance between the parking lots and the users' building, the number of users demanding parking for reserved and unreserved spaces from each building, and the number of available parking spaces in each parking lot. Associated with each dataset is the number of parking permits to be issued for each parking lot. Data were generated for the number of available parking spaces in each parking lot(see Table A.1, B.1, D.1 and C.1), the number of users demanding parking in each building (see Table A.2, B.2, D.2 and C.2) and the distance between each building and each parking lot (see Table A.3,

B.3, D.3 and C.3). However, the number of parking permits issued for each parking lot as shown in Table A.1, B.1, D.1 and C.1 were calculated as suggested by [57, 58] using Equation (4.1). Equation (4.1) must be satisfied in order to get equal probability for all users. Where p is the probability of a user bringing his car on a particular day and it is taken as 0.7 [57], $p.A_k$ is the mean of the distribution and $\sqrt{(p(1-p)A_k)}$ is the standard deviation of the distribution.

$$\frac{N_k - p.A_k}{\sqrt{(p(1-p)A_k)}} = \Psi \tag{4.1}$$

The value of Ψ in Equation (4.1) that gives the total number of parking permits issued, A_k, that will be equal to the number of users demanding parking, T_U, can be calculated by squaring Equation (4.1), rearranging the terms for the kth parking lot, ignoring small terms and then equating it to the total number of users demanding parking. The resulting equation is the quadratic equation in (4.2).

$$n\Psi^2 - 2\Psi \sum_{k=1}^{n} \sqrt{N_k} - 2(p.T_U - \sum_{k=1}^{n} N_k) = 0 \tag{4.2}$$

Solving the quadratic equation in (4.2), the value of Ψ is obtained and is used to get the number of the parking permits issued, A_k, for each parking lot in Equation (4.3).

$$A_k = \frac{(2N_k + \Psi^2) - 2\Psi\sqrt{(N_k)}}{2p} \tag{4.3}$$

Table 4.1 gives the number of buildings and parking lots in each dataset while a detailed history is found in appendix A - D.

Table 4.1: Different datasets

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Number of buildings	12	15	12	25
Number of parking lots	8	10	17	18

4.2 Experimental Settings

Experiments were performed on a PC using Windows XP 32-bit operating system with Intel(R) Core 2 Duo CPU @ 2.20 GHz, 0.98GB of RAM. This section gives a description of the dataset used for the experiment and the experimental setting for each of the algorithms that was implemented to solve the CPSA problem. The algorithms were in MATLAB codes.

4.2.1 GA Parameter Settings

The parameters were encoded as double vectors to reduce the computational overhead of encoding into binary strings and converting back into double values. A population size of 100 was used

to get a trade-off between speed and accuracy. The roulette wheel selection, which gives the good individuals the better chance to be selected than the weak ones based on the fitness of each individual, was used for the selection operator. To ensure that the best solution is not lost, the elite count of 2 was used. The heuristic crossover, which returns offspring of two parents that are at close distance to the parent with better fitness value than the parent with a worse fitness value, was used for the crossover operator with a crossover fraction of 0.8 as recommends in [27]. And to ensure feasible points are generated in the mutation operator, we used the adaptive feasible mutation.

4.2.2 PS Parameter Settings

The parameters used in PS are as follows: the initial mesh size was 1.0, mesh expansion value of 2 after each successful poll and mesh contraction value of 0.5 after each unsuccessful poll were used, no search method was specified and GPS positive basis 2N was used for the poll method which consists of 2N directions, where N is the number of independent variables.

4.2.3 PSwarm Parameter Settings

The following parameter settings were used to obtain solution for the PSwarm. The population size, called the swarm size was set to 40 as used in [36] while all other swarm parameters were as used in [47, 36]. The cognitial parameter, υ, and the social parameter, μ were set to 0.5, the initial weight and the final weight were linearly interpolated between 0.9 and 0.4.

4.3 Exact Results Using CPLEX Solver

4.3.1 Fitness Values across Datasets

The exact solution approach is well known for obtaining optimal solution to optimization problems when it is capable of solving such problems. The CPLEX solver was used to obtained the fitness values as shown in Figure 4.1. Dataset 1 has the fitness function value of 56747 with 49 constraints and 192 variables, dataset 2 has the fitness value of 61524 with 61 constraints and 300 variables, dataset 3 has the fitness value of 196500 with 76 constraints and 408 variables, and dataset 4 has the fitness value of 199480 with 105 constraints and 900 variables. Although the fitness values increase as the number of datasets increase for this study however, it is not dependent on the number of datasets but on the value of the distances between the parking lots and the buildings given in the datasets. If a given distance matrix has large numbers then the value of the fitness function will consequently be large.

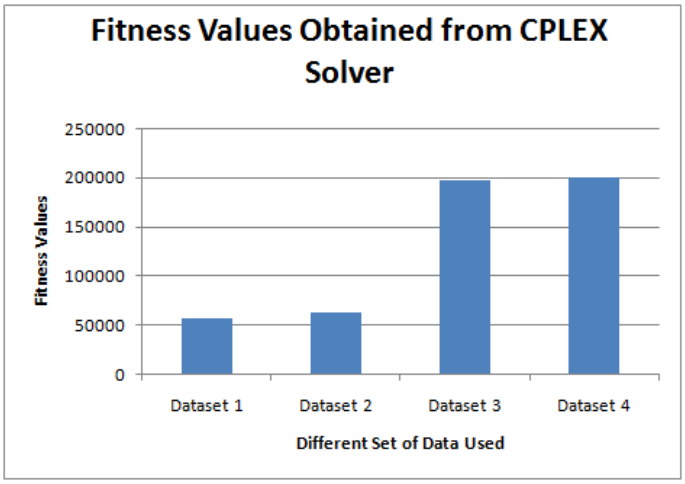

Figure 4.1: Fitness values obtained by CPLEX solver for the datasets

4.3.2 Distribution of Parking Spaces

This section presents the distribution of users to parking spaces as obtained by the CPLEX solver. The distribution gives the best minimization of the total distance walked by the users. The distance tables in Figure A.3, B.3, D.3 and C.3 will be used to evaluate how far or close a building is to a given parking lot. The word 'Rv' in the figure stands for reserved and the word 'UnRv' stands for unreserved. For each parking lot, the number of users allocated for reserved and unreserved parking spaces from each buildings are presented side by side. For each dataset, the consideration would be: (1) the number of buildings, in percentage, that have users that are allocated to parking lots around the three closest parking lots to them; (2) the number of buildings, in percentage, that have users that are partly allocated farther away and partly allocated within the three closest parking lots to them, either with a higher number of users farther away or a higher number of users closer by; and (3) the number of buildings in percentage that have no user allocated to any of the three closest parking lots.

4.3.2.1 Dataset 1

There are 8 parking lots and 12 buildings in dataset 1. This represents a fairly small campus situation case study. The data for this case study are given in appendix A, which has a total of 1267 users demanding for parking from all the buildings and a total of 1118 available parking spaces. The total number of reserved spaces on campus for this study is 391 while the total number of unreserved spaces is 727. Figure 4.2 gives the exact allocation for dataset 1. The distribution of the parking spaces to users from each building shown in Figure 4.2 suggests that 83% of the buildings has users allocated to three of the closest parking lots and 17% has users allocated to parking lots that are partly closer and partly far away.

Buildings	ParkingLot1		ParkingLot2		ParkingLot3		ParkingLot4		ParkingLot5		ParkingLot6		ParkingLot7		ParkingLot8	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1					27	65										
2							9	28								
3							5	15			22	51				
4											14	26			37	76
5							50	100								
6					7	6							36	83		
7			25	62												
8	25	50					5	20								
9			28	67												
10									33	75						
11			12	23	5	11									12	34
12									19	40			20	44		

Figure 4.2: Allocation done by CPLEX: Dataset 1

4.3.2.2 Dataset 2

Dataset 2 consists of 10 parking lots and 15 buildings. This represents a bigger campus than the case considered in dataset 1. The data in appendix B gives details of this. The task is to allocate 1345 users to 1018 available parking spaces. Figure 4.3 gives the exact allocation for dataset 2. For this dataset, the allocation distribution in Figure 4.3 suggests that 80% of the buildings has users allocated to three of the closest parking lots, 13% has users allocated to parking lots that are partly closer and partly far away, and 7% has users that are allocated to parking lots not close to their buildings.

Buildings	ParkingLot1		ParkingLot2		ParkingLot3		ParkingLot4		ParkingLot5		ParkingLot6		ParkingLot7		ParkingLot8		ParkingLot9		ParkingLot10	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1																			31	72
2	17	51																		
3							2	39							43	78				
4							32	73												
5	3	2							5	30										
6									13	13			26	76						
7											29	72								
8															2	35	5			
9			27	75															14	36
10							14	22											7	38
11											15	48								
12																	9	35		
13								2					13	31			7	22		
14									15	48										
15					27	75										12				

Figure 4.3: Allocation done by CPLEX: Dataset 2

4.3.2.3 Dataset 3

The case study of the campus considered in dataset 3 has a large number of parking lots than the number of buildings. For dataset 3, there are 17 parking lots and 12 buildings where users go after parking their cars. This could be seen in appendix C. Here, a total of 2541 users are to be allocated to 2025 parking spaces. Figure 4.4 and 4.5 give the exact solution for dataset 3. The distribution shown in Figure 4.4 and 4.5 indicates that 50% of the buildings has users allocated to three of the closest parking lots and 50% has users allocated to parking lots that are partly close and partly far away from the buildings.

Buildings	ParkingLot1		ParkingLot2		ParkingLot3		ParkingLot4		ParkingLot5		ParkingLot6		ParkingLot7		ParkingLot8		ParkingLot9		ParkingLot10	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1					38	97											22	53	3	
2													36	33					41	114
3													11	57						
4									4	19			3	42						
5							19	46												
6											43	110								
7	11	24														27				
8									35	118										
9																				
10			59	157																
11									13											
12															79	185				

Figure 4.4: Allocation done by CPLEX: Dataset 3

Buildings	ParkingLot11		ParkingLot12		ParkingLot13		ParkingLot14		ParkingLot15		ParkingLot16		ParkingLot17	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1			15	29		5								
2											33	84		
3													31	73
4							18	41						
5					1	45								
6	15	34											9	21
7													17	55
8														
9			13	57										
10			12	16										
11					44	86			62	164				
12					21	41								

Figure 4.5: Allocation done by CPLEX: Dataset 3 continued

4.3.2.4 Dataset 4

Dataset 4 has the highest number of buildings and parking lots used for this study which consists of 18 parking lots and 25 buildings. This case represents a fairly large campus situation. The problem is to allocate a total of 2334 users to 1979 available parking spaces. See appendix D for the details of other data associated with this case study. Figure 4.6 and 4.7 give the exact allocation for dataset 4. The distribution shown in Figure 4.6 and 4.7 suggest that 72% of the buildings has users allocated to three of the closest parking lots, 24% has users allocated to parking lots that are partly closer and partly far away, and 4% had users that are allocated to parking lots not close to their buildings.

Buildings	ParkingLot1		ParkingLot2		ParkingLot3		ParkingLot4		ParkingLot5		ParkingLot6		ParkingLot7		ParkingLot8		ParkingLot9		ParkingLot10	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1															14	40				
2											15	42								
3	6	15																	9	22
4					17	43														
5					21	56														
6																				
7			25	62																
8																				
9			2												22	39				
10												6								
11																				
12									11	34										
13									21	43										
14											1						44	92		
15															16	49				
16																			43	89
17							11	6										13		
18					3													1		
19									2											16
20													19	53						
21																				
22	31	72																		
23																				
24							15	50			15	22	17	25						
25														8						

Figure 4.6: Allocation done by CPLEX: Dataset 4

Buildings	ParkingLot11		ParkingLot12		ParkingLot13		ParkingLot14		ParkingLot15		ParkingLot16		ParkingLot17		ParkingLot18	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1																
2																
3					19	41					1					
4		6														
5																
6			25	61												
7										1						
8							2		3				19	61		
9													12	40		
10									12	35						
11									14	45						
12																
13								13								
14																
15																
16																
17															29	66
18							42	93								
19	26	50	11	23					10	9						
20																
21											20	55				
22																
23															21	56
24											1					
25											26	60	21	27		

Figure 4.7: Allocation done by CPLEX: Dataset 4 continued

4.4 Meta-heuristic Results Performances

4.4.1 Fitness Values across Datasets

The aim of this study was to determine how well the three meta-heuristic algorithms could solve the CPSA problem. The optimization model presented in Chapter 3 focused on the minimization of the total distance walked by the users. Hence, the comparison of the algorithms' performances are determined by how well they can minimize the fitness function defined in Equation (3.9). The results in Figure 4.8, 4.9 and 4.10 show the different performances of the algorithms for 10000, 20000 and 30000 number of iterations. The essence of comparing the performances over different number of iterations is to ascertain the consistency of the solution of the algorithms as the number of iterations increase.

The results indicate that an increase in the number of iterations would lead to an improvement in the fitness function values, which in turn minimizes the overall distance walked by the users. Taking the performances across the dataset used, it was observed that as the dataset increased in size, the fitness values for the PS and PSwarm algorithms tend to be closer to each other while that of the GA increased in its deviation from the other two algorithms. This is due to the fact that as the number of data increases in the dataset, the basic GA is incapable of efficiently solving the problem, leading to its inability to obtain results that are of close range to the other two algorithms.

In terms of minimizing the fitness function value, the results in Figure 4.8, 4.9 and 4.10 indicate that PSwarm outperforms the GA and PS algorithms. Since PSwarm is the only hybrid algorithm of the three algorithms, its superior performance suggests that hybrid algorithms could be more suitable for addressing the CPSA problem. Following the PSwarm algorithm in performance is the PS algorithm. The PS algorithm obtains good results which are close to the ones obtained by the PSwarm algorithm. It outperforms the standard GA, hence, it offers an improvement over the GA in terms of its allocation of users to various parking spaces. With regards to the performance of the GA, it is observed that the standard GA performs poorly compared to the PS and PSwarm algorithms. This actually shows the limitation of the standard GA when it is used to solve a constrained optimization problem that has a large amount of data.

4.4.2 Execution Time of the Algorithms for Different Datasets

Associated with the results compared, as extracted from the three algorithms, is the execution time. The essence of comparing their execution times is to identify which of the three algorithms is the fastest and which one is the slowest to converge. The results in Figure 4.11, 4.12 and 4.13 show the execution time taken by each algorithm across the dataset for the number of iterations of 10000, 20000 and 30000. The results indicate that the execution time increases as the number of data increases. Since the number of parking lots and buildings on campus increase as the number

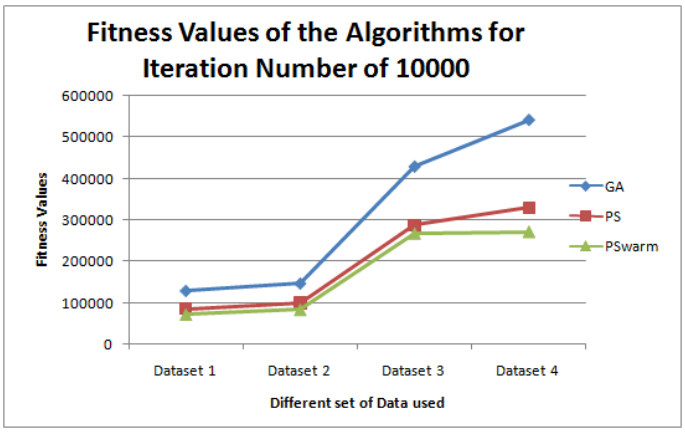

Figure 4.8: Comparison of the objective values for 10000 iterations

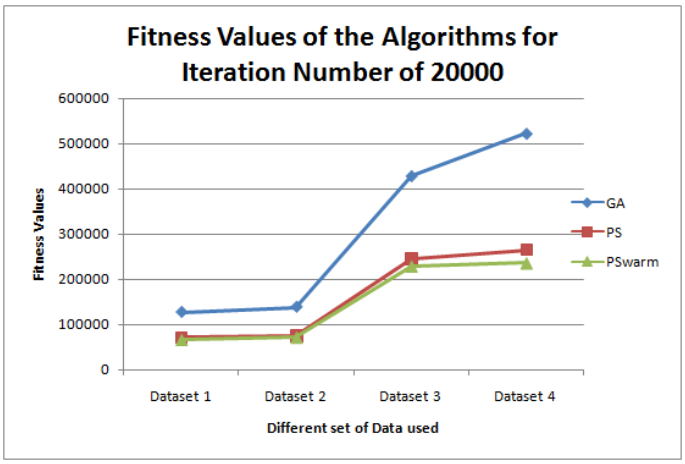

Figure 4.9: Comparison of the objective values for 20000 iterations

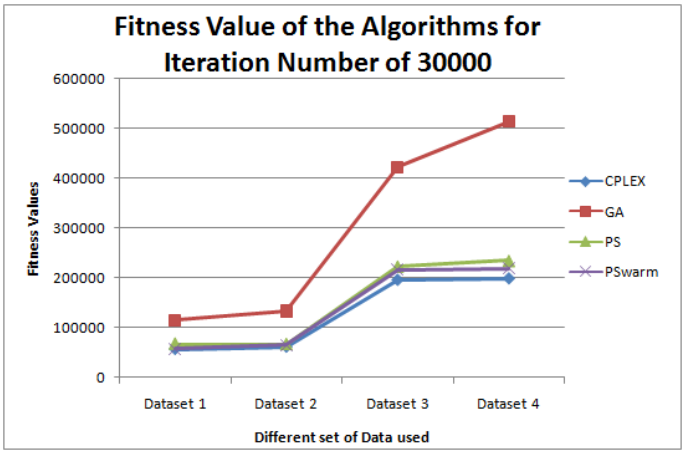

Figure 4.10: Comparison of the objective values for 30000 iterations

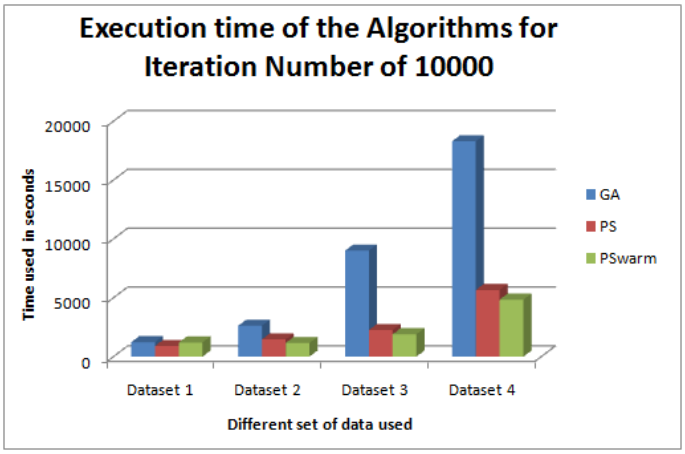

Figure 4.11: Comparison of the execution time for 10000 iterations

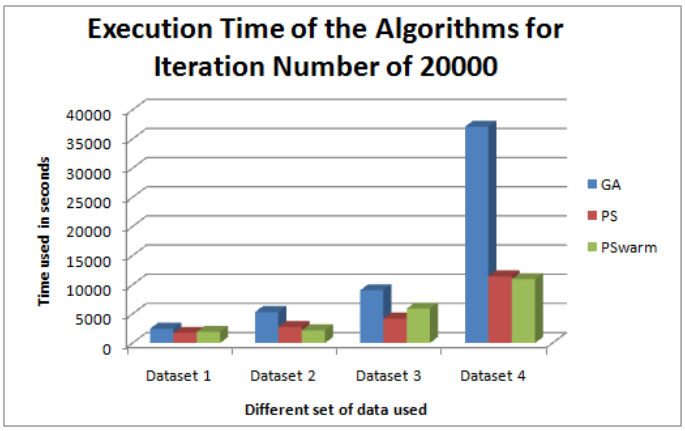

Figure 4.12: Comparison of the execution time for 20000 iterations

of each dataset increases, as shown in Table 4.1, the function evaluations of the algorithms take longer to complete for large number of datasets compared to the smaller ones.

The results also indicate that the PSwarm algorithm was the fastest overall, this makes the PSawrm algorithm better than the other two algorithms in terms of efficiency and quality of the solution as discussed in the previous section. Although, the PSwarm algorithm is the fastest of the three, it can be observed in Figure 4.11 that the PS algorithm was faster than the PSwarm algorithm in dataset 1; and in Figure 4.12, it is also faster than the PSwarm algorithm in dataset 1 and 2. On the other hand, GA is the slowest of the three algorithms for each number of iterations in all the datasets under consideration. This is attributed to the number of computational operations that are usually performed by the GA operators.

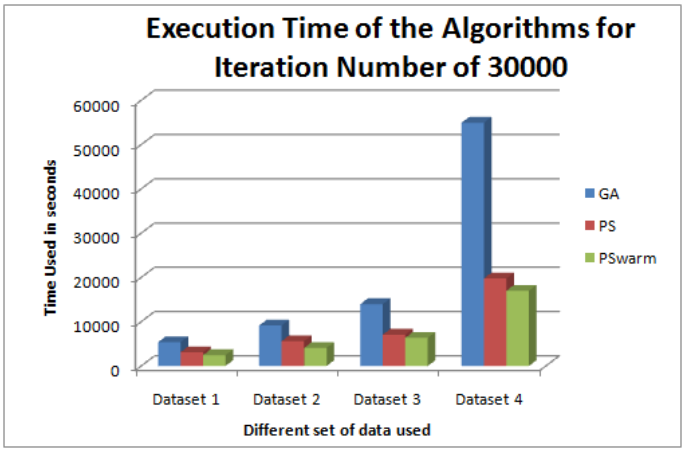

Figure 4.13: Comparison of the execution time for 30000 iterations

4.4.3 Distribution of Parking Spaces

The comparison done earlier was in terms of the fitness function values and execution time taken by the three algorithms. However, the fitness function value is the value of the total distance minimized. Hence, it is a function of the allocation done by these algorithms. The allocation solution by the algorithms for each dataset is presented in this section. The discussion of the various allocations will centre on the three closest parking lots to any building of interest. Since the objective is to minimize the total distance walked by the users to their various buildings, it is expected that the algorithms would allocate parking spaces that are as close as possible to the users' buildings. Hence, the three closest parking lots to each building and how the algorithms allocate users from the buildings to these parking lots would be considered. This will help to know how optimal or close to optimal the allocations are.

4.4.3.1 Dataset 1

The allocation distribution by each algorithm for dataset 1 is shown in Figure 4.14 for GA, Figure 4.15 for PS and Figure 4.16 for PSwarm. Comparing the distributions of the algorithms in terms of the number of buildings that have users allocated to closer parking lots or not, it was observed that in the GA allocation distribution (Figure 4.14), 100% of the buildings has users allocated to parking lots that are partly close and partly far from their buildings with most of the users allocated to parking lots that are far from their buildings than the closer ones. In the PS allocation distribution (Figure 4.15), 42% of the buildings has users allocated to parking spaces within the three closest parking lots and 58% has users allocated to parking lots that are partly close and partly far from their buildings. While in the PSwarm allocation distribution (Figure 4.16), 83% of the buildings has users allocated to parking lots within the three closest parking lots and 17% has users allocated to parking lots that are partly close and partly far from their buildings. The allocations suggest

Buildings	ParkingLot1		ParkingLot2		ParkingLot3		ParkingLot4		ParkingLot5		ParkingLot6		ParkingLot7		ParkingLot8	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1	6	2	2	7	3	12	6	8	3	12	2	10	1	8	4	6
2		1	4	5		2	3	9	1				1	5		6
3			3	14	5	5	7	18	7	13	4	5		6	1	5
4	13	12	5	21	4	12		15	4	4	5	11	8	9	12	18
5		3	4	10	3	8	15	23	3	15	6	10	13	21	6	10
6	1	9	6	5	13	8	1	16	6	8	2		9	25	5	18
7			10	17	1	13	1	3	2	9		9	5	5	6	6
8	2	8		10			16	15	2	10		8	10	15		4
9			7	6		3	11	16	5	15	1	6	2	13	2	8
10	1	7	11	19	3	11		8	9	14	1	4	6	3	2	9
11		3	6	16	5	4		14		8	9	8		8	9	7
12	2	5	7	22	2	4	9	18	10	7	6	6	1	9	2	13

Figure 4.14: Allocation done by GA: Dataset 1

Buildings	ParkingLot1		ParkingLot2		ParkingLot3		ParkingLot4		ParkingLot5		ParkingLot6		ParkingLot7		ParkingLot8	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1					27	65										
2							9	28								
3					6		1	65			11			1		
4	18										25	41			8	61
5							50	31		69						
6					6				16				4	87	17	2
7			25	6		17								39		
8	7	50						20					23			
9			28	31				19		17						
10									33	28						47
11			5	68											24	
12			7	47					3	1		36	29			

Figure 4.15: Allocation done by PS: Dataset 1

that PSwarm minimizes better the total distance walked by the users compared to the PS and GA.

4.4.3.2 Dataset 2

Figure 4.17, 4.18 and 4.19 give the allocation distribution by the three algorithms for dataset 2. In the GA allocation distribution (Figure 4.17), the percentage of buildings which has users partly allocated to parking lots that are close and far from the buildings is still 100% with the number of those that are far being higher than the number of those that are close in most cases. In the PS allocation distribution (Figure 4.18) 53% of the buildings has users allocated to parking spaces within the three closest parking lots with more allocations being made to the second and third closest parking lots, 40% has users partly allocated to parking lots that were closer and partly allocated to farther parking lots, and 7% has users allocated to parking lots that are not within the closest parking lots. While in the PSwarm allocation distribution (Figure 4.19), 47% of the buildings has users allocated to one of the three closest parking lots with more allocations to the

Buildings	ParkingLot1		ParkingLot2		ParkingLot3		ParkingLot4		ParkingLot5		ParkingLot6		ParkingLot7		ParkingLot8	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1					27	65										
2							9	28								
3							9	34			18	32				
4	4	28									18	45			29	29
5							50	100								
6					12								31	35		54
7			25	48		14										
8	21	22					1						8	48		
9			28	66				1								
10			3						30	75						
11			9	38		3									20	27
12									22	40			17	44		

Figure 4.16: Allocation done by PSwarm: Dataset 1

first and second closest parking lots, 46% has users allocated to parking lot that are partly close and partly far from their buildings with a higher number to the closer parking lots and 7% has users allocated to parking lots outside the three closest parking lots. The PSwarm algorithm minimizes the total distance walked by the users better than the PS and GA.

Buildings	ParkingLot1		ParkingLot2		ParkingLot3		ParkingLot4		ParkingLot5		ParkingLot6		ParkingLot7		ParkingLot8		ParkingLot9		ParkingLot10	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1	5		1	2	1		4	9	6		9	5	1	45	1	8	1		2	3
2	2	4			1	1	1	4		27	2	6	1	1			1	6	9	2
3	1	6	2	3	2	39	7	13	12		1	4	1	28	7	7	12	16		1
4	1	11			1		9	55	1	1	9	1	2		8	1			1	4
5	1	1			4				2	5		1			1	19		1		5
6	4	3	12	23		25	1	5	3	18	4		12	9	2	4	1	2		
7	4	6	2	3	2	3	14			32	6	12				9	1	7		
8								3			1	26	1		5	5		1		
9		22	2	33		1			1	2	7	1				6		3	31	43
10	1		1			1	3	1	4		1		9	1		53			2	4
11	1				8	2		42				2			6	2				
12				1	1		1	3		2		6	5	18	1			7	1	2
13			1	1			6	1			1	46	6	5	5		1	2		
14				9	2		1		4	3		14			4	9	2	12	2	1
15			6		5	3	1			1	3		1		5	2	2		4	81

Figure 4.17: Allocation done by GA: Dataset 2

Buildings	ParkingLot1		ParkingLot2		ParkingLot3		ParkingLot4		ParkingLot5		ParkingLot6		ParkingLot7		ParkingLot8		ParkingLot9		ParkingLot10	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1								25											31	47
2							5										12	51		
3					18		19	41							8	76				
4	20	26			4		8	34				13								
5		17			5				3	14						1				
6									12	26			19	62	8	1				
7				6							29	66								
8							5	9							2	26				
9			27	69					3	3									11	39
10							11												10	60
11		10				9					15	29								
12												16		17			9	6		
13								27					20	28						
14									15	48										
15						66									27	21				

Figure 4.18: Allocation done by PS: Dataset 2

Buildings	ParkingLot1		ParkingLot2		ParkingLot3		ParkingLot4		ParkingLot5		ParkingLot6		ParkingLot7		ParkingLot8		ParkingLot9		ParkingLot10	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1							3	38											28	34
2	14	27									3							24		
3			3				4	27							38	90				
4	2						30	27				31								15
5		21							8	11										
6									23	33			16	56						
7				41							29	31								
8															4	9	3	26		
9			24	34															17	77
10							11	40					3						7	20
11	4	5			3						8	29				14				
12											4	33					5	6		
13								4					20	51						
14									2	47							13	1		
15					24	75									3	12				

Figure 4.19: Allocation done by PSwarm: Dataset 2

4.4.3.3 Dataset 3

The results of the allocation by the algorithms for dataset 3 can be seen in Figure 4.20 and 4.21 for the GA, Figures 4.22 and 4.23 for PS, and Figures 4.24 and 4.25 for PSwarm. The findings for the various allocations are as follows: in the GA allocation distribution (Figures 4.20 and 4.21), 100% of the buildings still has users allocated to parking lots that are partly close and partly far from their buildings with more of the users being allocated to the far parking lots. In the PS allocation distribution (Figures 4.22 and 4.23), 8% of the buildings has users allocated within the three closest parking lots and 92% has users allocated to parking lots that partly close and partly far from their buildings. While in the PSwarm allocation distribution (Figures 4.24 and 4.25), 17% of the buildings has users that are allocated to the three closest parking lots and 83% has users that are allocated to parking lots that are partly close and partly far from their buildings. For this dataset, PSwarm algorithm minimizes the total distance walked by the users better than PS and GA.

Buildings	ParkingLot1		ParkingLot2		ParkingLot3		ParkingLot4		ParkingLot5		ParkingLot6		ParkingLot7		ParkingLot8		ParkingLot9		ParkingLot10	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1			3	20	4	7	1	1	9	14	4	12	9	18	3	15	2	15	2	11
2		7	7	24	7	17	3	4	2	10	1	20	4	15	10	23	6	8	13	7
3				10		1		3	7	14	6	6	6	12	11	20	1	2		8
4			5	8		10			7	8	4	6		14		16				4
5			3	13				1		4	1	1		12	2	15		4		3
6			5	14	11	10	2	10	1	10	7	8	3	13		13	1	4	5	16
7		1	7	5		4	1	8	2	10	1	6	1	6	9	17	1	3		
8		3	1	18	2	6	1	7	7	6	8	2	5	1	4	12		1		9
9	1			1		4				4			2	1	3	7		1	2	4
10		4	2	12	1	11	3		9	10	1	12	6	14	11	20	2	5	1	21
11	7	9	19	19	11	20	2	4	5	24	5	23	6	16	18	25		8	9	8
12	3		7	13	2	7	6	8	3	23	5	14	8	10	8	29	9	2	12	23

Figure 4.20: Allocation done by GA: Dataset 3

Buildings	ParkingLot11		ParkingLot12		ParkingLot13		ParkingLot14		ParkingLot15		ParkingLot16		ParkingLot17	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1		1	9	9		22		3	16	22	3	5	13	9
2	13	2	6	19	7	19	1	13	9	16	14	9	7	18
3		10		13	2	9				16	1		8	6
4		5		1	4	7	4	1				10	1	12
5		3	4	8	6	10		6	4			9		2
6		1	11	11	14	15	1	1	4	18		7	2	14
7				10	5	12				11		1	1	12
8				2		7	3		2	25	1	6	1	13
9		1	1	3		7			2	21			2	3
10		1	2	7	10	23		2	14	11	5	4	4	16
11			7	6	11	28	8	11	3	11	2	19	6	19
12	2	10		13	7	18	1	4	8	13	7	14	12	25

Figure 4.21: Allocation done by GA: Dataset 3 continued

Buildings	ParkingLot1		ParkingLot2		ParkingLot3		ParkingLot4		ParkingLot5		ParkingLot6		ParkingLot7		ParkingLot8		ParkingLot9		ParkingLot10	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1					38	40											10	53	2	84
2				77							8		40	16					42	30
3		1									19		10	115						
4									7	46				1						
5																				
6											11									
7	11	23							4							33	12			
8						6			30	2	5	110								
9									3	54										
10			59	80																
11						51	19	46	2	35						48				
12									6						79	131				

Figure 4.22: Allocation done by PS: Dataset 3

Buildings	ParkingLot11		ParkingLot12		ParkingLot13		ParkingLot14		ParkingLot15		ParkingLot16		ParkingLot17	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1			28	7										
2		32									20	76		
3											12	8	1	6
4						14	18	41						
5					19					87	1			4
6		1								25			56	139
7					1					50				
8														
9		1		2	10									
10			12	93										
11	15				21	68			62	2				
12					15	95								

Figure 4.23: Allocation done by PS: Dataset 3 continued

Buildings	ParkingLot1		ParkingLot2		ParkingLot3		ParkingLot4		ParkingLot5		ParkingLot6		ParkingLot7		ParkingLot8		ParkingLot9		ParkingLot10	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1					27											63	22	42		
2				12									41	17					42	114
3													9	2					2	
4									7	43				18						
5	3						9	32												
6								4			43	70								
7	8	16														62		11		
8					3	97			32	20										
9									11											
10			59	145								28								
11		8			8		10	10	2	41		12		95						
12										33					79	87				

Figure 4.24: Allocation done by PSwarm: Dataset 3

Buildings	ParkingLot11		ParkingLot12		ParkingLot13		ParkingLot14		ParkingLot15		ParkingLot16		ParkingLot17	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1			26	45	3	34								
2										4	27	84		
3											6		25	128
4							18	41						
5					8	59								
6	15	19								60			9	12
7						4			19	5			1	8
8														1
9			2	57										
10			12											
11					34				43	84			22	
12		15			21	80				11				

Figure 4.25: Allocation done by PSwarm: Dataset 3 continued

4.4.3.4 Dataset 4

The allocations for this case are shown in Figures 4.26 and 4.27 for GA solution, Figures 4.28 and 4.29 for PS algorithm, and Figures 4.30 and 4.31 for PSwarm algorithm. In the GA allocation distribution (Figures 4.26 and 4.27), which is still the same as observed in other datasets, 100% of the buildings has users allocated to parking lots that are partly close and partly far from the buildings with a good number of them allocated to the far parking lots. In the PS allocation distribution (Figures 4.28 and 4.29), 16% of the buildings has users allocated to parking spaces within the three parking lots and 84% has users allocated to parking lots that are partly close and partly far from the buildings. While in the PSwarm allocation distribution (Figures 4.30 and 4.31), 40% of the buildings has users allocated to parking lots within the three closest parking lots, 56% has users allocated to parking lots that are partly close and far from the buildings, and 4% has users that were not allocated to any of the three closest parking lots. The PSwarm algorithm minimizes the total distance walked by the users better than the PS and GA in this dataset.

Buildings	ParkingLot1		ParkingLot2		ParkingLot3		ParkingLot4		ParkingLot5		ParkingLot6		ParkingLot7		ParkingLot8		ParkingLot9		ParkingLot10	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1	2		1	2	1	2				1		4	1	2	1	3		5		5
2	1	8		1	1	2		2	1	1			1			3	3	5	2	1
3		3	3	2	2	4	2		2	7	2		2	2		5	3	5		5
4		2		5	1	6	1	6	2	1		2	1	3	3	7	2	2	1	1
5				3	3	1			1	5			3	1	1	1	3	3	3	7
6		1	1		1	3	4	6	1	1		2		4	2	6		3	6	4
7	4	6		3	3	4			3	2		3		3		8		1	3	7
8	2	6	2	1	6	7	2	1			2	2	1	4		6	2	5	1	3
9		7		1	4	6	2	3	1	4		9		3	5	5	5	2	5	5
10		3		2	5	3			1				1	3		3				3
11	1	2		1	1	4			2	2	2	4		2	2	6		5	2	2
12					2	2		1		1					2	2		6		7
13		1	2	5		1		2		2		1	2	5	3	4	4	1	1	6
14	2	4			1	5	3	3	3	3	2	1	6	6	5	9	2	8	1	9
15		4		1		7		5		5	1			4		4	1	2		1
16	7	5	3	4		5		5	1	2		6	6	6	2	3	1	5	1	5
17	3	5	2	4		8		2	1	2	3	4	5	8	1	6	3	7	3	4
18	4	2	2	8	1	2	5	4	1	7	1	2	2	1	5	3	2	4	3	7
19	1	6	3	7		6	1	1	4	6	2	8		2	5	5	5	12	5	13
20	1	2	3	2	3	3			1	3		5		1		3	2	4	2	2
21		6		3		1		2	3	3		3	1	4		4		4		6
22	3	2	1	2	2	4		4	2		2	5		6	5	6		2	3	5
23	2		1			2	1	4	2	6	6	3		4	3	4		4		5
24	3	7	1	4	3	4	1	2	1	8	6	2	2	5	3	13	3	7	5	11
25	1	5	2	1	1	7	4	3	1	5	2	4	2	7	4	9	3	4	5	3

Figure 4.26: Allocation done by GA: Dataset 4

Buildings	ParkingLot11		ParkingLot12		ParkingLot13		ParkingLot14		ParkingLot15		ParkingLot16		ParkingLot17		ParkingLot18	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1	3			3		2				3	1	2	2		2	6
2				5		1		3	1	2		5	3		2	3
3	2	1	1	5	1	2	2	8	6	8	4	9	3	7		5
4	1		1	4	1			1		1		2	2	4	1	2
5	1		1	3			2	7	2	7		6	1	4		8
6	1		2	6	3	6		2		4	2	6	2	1		6
7		4	1	5		1	1	2	3	2	2	5		4	5	3
8	1	2	1	2		3	1	5	2	3		1	1	5		5
9	1	4	3	2	1		1	3		11	5	5	2	6	1	3
10	1	3				6	1	3		2		2	3	5		3
11		2	2			3		4	2			2		2		4
12				2	3		1	1	1	2		5	1	3	1	2
13		1	1	4	1		2	6		4	2	7	2	1	1	5
14	1	3	6	6	1	3	2	3	3	2		4	2	17	5	6
15	3	2			1	1		3		3	4	1	2	6	4	
16	1	7	2	6		2	8	5	1	7	1	8	6	4	3	4
17		1	1	7	2		5	5	3	1	2	8	3	7	3	6
18	1	2		4	1	4	4	11	6	3		12	5	5	2	13
19	1	2	5	1		3	3	7	1	3	5	4	2	7	6	5
20			2			1		2	2	5	1	1	2	9		10
21		6	2	1	2		1	1		2	5	3	1	2	5	4
22	5	4	1	3		2		5		6	3	2	2	7	2	7
23	1	1	1	3		1	1	7	1	1	1	2	1	5		4
24	2	1	2	5	1		6	7	2	3	2	6	1	6	4	6
25		10	1	7	1		3	5	3	5	8	7	3	11	3	2

Figure 4.27: Allocation done by GA: Dataset 4 continued

Buildings	ParkingLot1		ParkingLot2		ParkingLot3		ParkingLot4		ParkingLot5		ParkingLot6		ParkingLot7		ParkingLot8		ParkingLot9		ParkingLot10	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1										21					7	15				
2											3	36								
3																			17	61
4												10							10	
5					3	51			3											
6		26																		
7				33											7	1				
8																				
9			27	14							7				2	63				
10				14										27						
11												8			14	12				
12									11	34										
13									7	12										
14	6							35									19	57		
15				1								10		1	3	37				
16		1															25	49		38
17								21												
18					38	3						5								
19															19				14	28
20						45							14	8						
21										10	20	1								
22	31	60																		
23									13		1									
24							26						8	50					11	
25													14							

Figure 4.28: Allocation done by PS: Dataset 4

Buildings	ParkingLot11		ParkingLot12		ParkingLot13		ParkingLot14		ParkingLot15		ParkingLot16		ParkingLot17		ParkingLot18	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1											7	3				1
2													12	6		
3											18	17				
4	3	39			4											
5			15	5												
6			3	35							22					
7									18	29						
8									16	15			8	46		
9														2		
10					12											
11										25						
12																
13							14	44								
14							20									
15	13															
16															18	1
17	8		8												24	64
18							7	62						24		
19	2	17	10			41			4	9				3		
20							3								2	
21				44												
22										12						
23									1						6	56
24											1		2	47		
25					3							95	30			

Figure 4.29: Allocation done by PS: Dataset 4 continued

Buildings	ParkingLot1		ParkingLot2		ParkingLot3		ParkingLot4		ParkingLot5		ParkingLot6		ParkingLot7		ParkingLot8		ParkingLot9		ParkingLot10	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1															14	40				
2											13	25		11						
3	15																		8	27
4					8	17														
5			4		13	30			4	9										
6																				
7			7	9											11	5				
8																				
9				25											11	20				
10			8	17																
11																28				
12									11	34										
13									16	30										
14							1										44	92		
15															16	34				
16		31											1					7	4	29
17							8									1		7		
18					10	49														
19										4									40	71
20			8	11	10	2							1	39						
21																				
22	22	56										16								
23									3		17									
24						1	17	56					3							
25											1	29	31	36						

Figure 4.30: Allocation done by PSwarm: Dataset 4

Buildings	ParkingLot11		ParkingLot12		ParkingLot13		ParkingLot14		ParkingLot15		ParkingLot16		ParkingLot17		ParkingLot18	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
1																
2													2	6		
3					5	28					6	23	1			
4	4	20			5	12										
5				17												
6			25	61												
7									7	49						
8							4	45	10	8			10	8		
9													25	34		
10									4	16						8
11	6								8	17						
12																
13							5	16				10				
14																
15																15
16		15	5												33	7
17	15	4							1						16	73
18							35	45								
19		17	6	6									3			
20						1										
21											20	55				
22									9							
23														40	1	16
24											17		11	40		
25	1				9						5	27				3

Figure 4.31: Allocation done by PSwarm: Dataset 4 continued

4.5 Meta-heuristics Comparison with Exact Results

This section compares the results obtained from the meta-heuristic algorithms with the ones obtained from the CPLEX solver in terms of their fitness function values. The result in Figure 4.32 suggests that the solutions of the PSwarm and PS algorithms are closer to the best than those of the GA. Since the solutions obtained by the CPLEX solver give an exact solution which is the best, it can be observed that the algorithms, especially the PSwarm and PS algorithms performed well in addressing the CPSA problem. The fitness values for the PSwarm and PS algorithms are much closer to the fitness values of the CPLEX solver for dataset 1 and 2 than those obtained in datasets 3 and 4. However, as earlier stated, the solutions obtained by the GA for each dataset are farther from the best solution than the solutions obtained from the other two algorithms.

Percentage Relative Error

An error is a measure of the accuracy of the solution obtained in an experiment. The error in this study will be the meta-heuristic objective function value minus the best value obtained from the exact method. While the relative error will be the error divided by the best value. The formula in Equation (4.4) was used to obtain the percentage relative error of the heuristics solution for the dataset as seen in Table 4.2.

$$\%Error = \frac{(HeuristicsSolution - BestSolution)}{BestSolution} * 100\% \tag{4.4}$$

Table 4.2: Percentage relative error

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
GA	103%	117%	115%	158%
PS	20%	9%	14%	18%
PSwarm	4%	6%	10%	9%

Table 4.2 suggests that the percentage error obtained in the standard GA for all the datasets was greater than 100, this is a source of concern. The percentage error for PSwarm only falls between 0 - 10, while that of PS falls between 0 - 20.

The exact solution was expanded and tested with more larger datasets and the observations noticed from the experiments is as follows. The details of the additional datasets are as shown in Table 4.3, with the first one having a total number of 50 buildings and 40 parking lots, and the last one having a total number of 1000 buildings and 900 parking lots. It was observed that as the size of the dataset increases, the time taken by CPLEX to solve the problem also increases (see Figure 4.33). However, the CPLEX solver in AIMMS failed to generate a result when the number of parking lots increases to 900 and the number of buildings increases to 1000, hence, the proposed use of

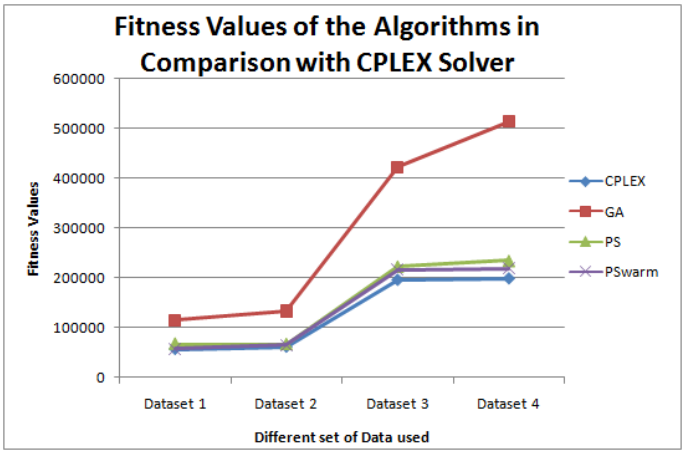

Figure 4.32: Comparing the fitness values with the CPLEX solver

meta-heuristics especially the PSwarm algorithm for the CPSA problem. The comparison studies of how close the heuristics are to the best solution are essential for further complex modeling of the CPSA problem which invariably will have no solution using the exact methods.

Table 4.3: Additional datasets

	Dataset 5	Dataset 6	Dataset 7	Dataset 8	Dataset 9
Number of buildings	50	100	250	550	1000
Number of parking lots	40	80	200	500	900

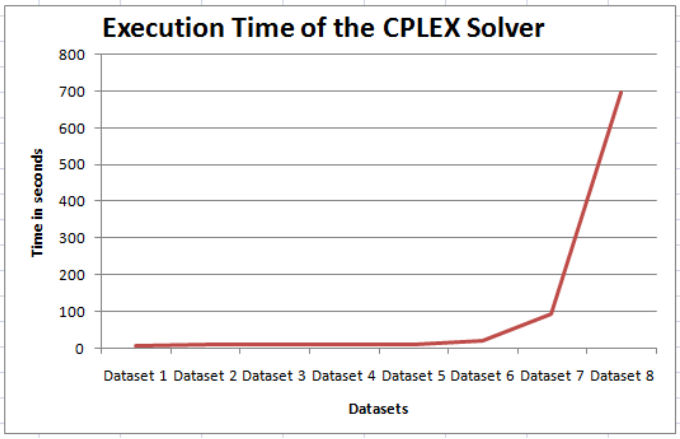

Figure 4.33: Execution time of the CPLEX solver for each dataset

4.6 Conclusion

This chapter describes the random variants of real life datasets that were used in the study and how they were generated. The machine and each algorithm settings in obtaining the solutions for the CPSA were also given. Following this, the fitness function values and allocation distributions of users to parking spaces for each dataset was obtained using the CPLEX solver. This was later used to calculate the percentage relative error for the three algorithms. The results obtained from the three algorithms were presented and compared in terms of their fitness function values, time taken, and the allocation distribution that was done for each building. It was observed that the PSwarm outperformed the PS and GA algorithms, perhaps due to the combined abilities of the PSO and PS algorithms. The PSwarm algorithm was the fastest overall. The GA performed poorly and was the slowest of the three algorithms.

Chapter 5

Conclusion and Future Research

> "All progress is precarious, and the solution of one problem brings us face to face with another problem ", [60].

5.1 Introduction

This chapter embodies the concluding remarks for this thesis. The research work addresses the CPSA problem. The problem is how to efficiently allocate parking spaces to different types of users in order to minimize the distance walked by each user. Two of the key objectives of this study are to formulate an optimization model for the parking allocation problem and to proffer different approaches to its solution.
Section 5.2 presents how each objective of the study is addressed in the various chapters of the study. Section 5.3 gives a summary of the research work. Section 5.4 gives possible directions that future research could take. Section 5.5 contains the conclusion of this chapter and the significance of the study.

5.2 Organisation of Objectives

This study has seven objectives as discussed in section 1.4 that give an overview of the work done in this study. Figure 5.1 shows how each of the objectives of the study is connected to each chapter in the study. Objective 1 is to study the existing CPSA problem which is covered in chapters 1 and 3. Objective 2 is to formulate the problem as a COP which is covered in chapters 2 and 3. Objective 3 is to develop an appropriate model for solving the problem which is presented in chapter 3. Objective 4 is to apply some heuristic algorithms to solve the problem which is described in chapters 2 and 3. Objective 5 is to compare the results obtained from the heuristics with that obtained from the CPLEX software discussed in chapter 4. Objective 6 is the analysis of

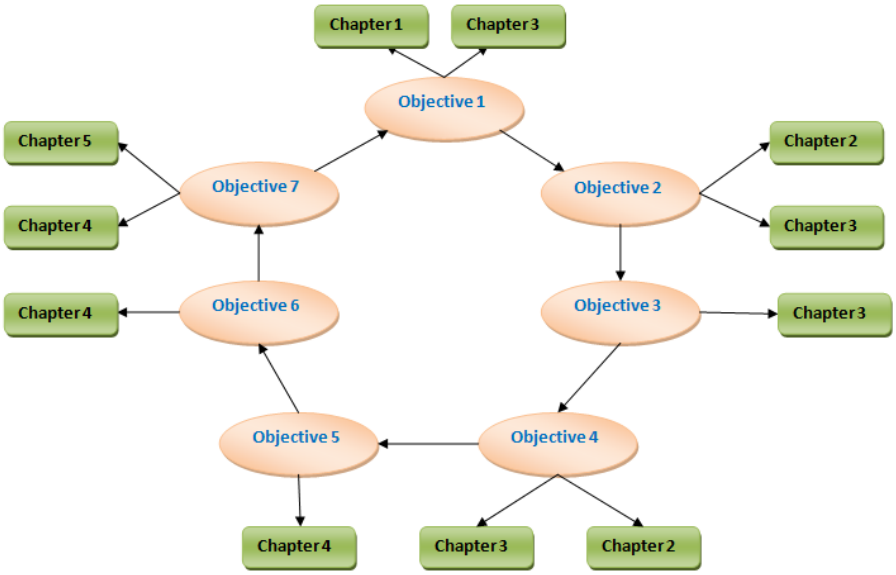

Figure 5.1: Connecting the objectives with each chapter

the performance of the techniques which is presented in chapter 4. Finally, objective 7 is to draw valuable conclusions from the results obtained in chapters 4 and 5.

5.3 Summary of Research Study

An investigation of the mathematical model formulated in [58] was done and an improvement of the constraints to cater for the parking reserved policy in the university environment was made. The resulting model was more suited for the parking allocation on campus for both reserved and unreserved parking spaces. Some variants of real world data were generated and were used to evaluate the optimization model. The datasets ranged from parking data for a fairly small campus to parking data for a fairly large campus. Following this, the optimization model was implemented using the CPLEX solver to obtain exact solutions for each dataset. The model was also implemented using three meta-heuristic algorithms which are the GA, PS and PSwarm. The meta-heuristic algorithms used were in their simple form.

The results obtained from these algorithms were compared, this is because previous researches or literature along this particular line of study are not available for comparison, in terms of the value of the fitness functions, execution times and the allocation output for the different datasets. The results obtained for the three meta-heuristic algorithms were also compared to the results obtained from the CPLEX solver. The results obtained using these datasets indicate that the meta-heuristic algorithms can successfully solve the CPSA problem and give solutions that are near optimal. It was observed that the PSwarm had the best solutions of the three algorithms and

that it is the fastest overall. This can be partly due to the fact that PSwarm is the only hybrid algorithm of the three algorithms and partly because the PSO that is hybridised with the PS possesses a higher capability of finding global optimal. Following the PSwarm algorithm is the PS algorithm which attained some results that are close to those obtained by the PSwarm algorithm. However, the GA could not outperform the other algorithms in this study.

5.4 Future Research

The optimization model for the CPSA can be extended further to accommodate multiple types of users either for reserved or unreserved parking such as the short term parkers, long term parkers and visitors . Some additional constraints to address this can be included in the model. Other possible objectives, such as maximizing the revenue generated by the university or maximizing parking usage can also be considered. The model can be formulated as a multi-objective optimization model considering simultaneously more than one possible objective subject to the same set of constraints or with additional constraints. It is believed that the quality of the solution obtained as well as the speed of the GA can be improved upon. The basic GA used in this study could be hybridized with other heuristic algorithms such as HC, Tabu Search(TS) and SA for significant improvements. Further parameter tuning can be examined for both the PS and PSwarm algorithms to improve the quality of their solutions.

The solution approaches outlined in this study can be developed into a standalone software application for ease of entry of users' information and for better parking space distribution for the campus parking managers.The solution approaches can be embedded in a web application software that allows users enter their details from the comfort of their offices and get their parking allocation immediately.

5.5 Conclusion

This study is important for campus parking managers and traffic planners, such as the Risk Management Services (RMS) at the University of KwaZulu-Natal, and for other universities. The study will help parking managers and traffic planners efficiently allocate parking spaces to users which in turns saves time and increases the productivity of research and academic work by minimizing the distance walked by the users from their buildings to the parking lots and vice versa. It will also help in the determination of the number of parking permits to issue for each parking lot based on the number of users demanding for parking, this avoids indiscriminate allocation of parking permits and decreases overcrowding at particular parking lots while other parking lots (often those that are far away) are less crowded.

List of References

[1] Abramson M. A. *Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems*. PhD thesis, Department of Computational and Applied Mathematics, Rice University, 2002.

[2] Batabyal A. A and Nijkamp P. A probabilistic analysis of two university parking issues. *The Annals of Regional Science*, 44(1):111–120, 2010.

[3] Charles A. and Dennis Jr. J. E. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903, 2002.

[4] Hirabayashi A., Aranha C., and Iba H. Optimization of the trading rule in foreign exchange using genetic algorithm. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1529–1536. ACM, 2009.

[5] Homaifar A., Qi C. X., and Lai S. H. Constrained optimization via genetic algorithms. *SIMULATION*, 62(4):242–253, 1994.

[6] Hulme-Moir A. *Making Way for the Car: Minimum Parking Requirements and Porirua City Centre*. Victoria University of Wellington, 2010.

[7] Narragon E. A., Dessouky M. I., and DeVor R. E. A probabilistic model for analyzing campus parking policies. *Operations research*, pages 1025–1039, 1974.

[8] Sarker R. A. and Newton C. *Optimization Modelling: A practical approach*. 2008.

[9] AIMMS. Aimms @ONLINE. http://business.aimms.com/. Accessed: 2012-01-30.

[10] Hadj alouane A. B. and Bean J. C. A genetic algorithm for the multiple-choice integer program. *Operations Research*, 1992.

[11] Grimbleby J. B. Automatic analogue circuit synthesis using genetic algorithms. *Circuits, Devices and Systems, IEE Proceedings*, 147(6):319–323, 2000.

[12] Andrej C. Topics in applied math: Methods of optimization @ONLINE. http://www.math.utah.edu/ cherk/teach/opt/course.html. Accessed: 2011-12-08.

[13] Blum C. and Roli A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.

[14] Chakraborty R. C. Artificial intelligence @ONLINE. http://www.myreaders.info/html/artificial-intelligence.html. Accessed: 2012-10-09.

[15] Coello C. A. C. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, 2002.

[16] Davis C. Theory of positive linear dependence. *American Journal of Mathematics*, 76:733–746, 1954.

[17] Mouskos K. C., Tvantzis J., Bernstein D., and Sansil A. Mathematical formulation of a deterministic parking reservation system(prs) with fixed costs. In *Electrotechnical Conference, 2000. MELECON 2000. 10th Mediterranean*, volume 2, pages 648–651. IEEE, 2000.

[18] Doval D., Mancoridis S., and Mitchell B. S. Automatic clustering of software systems using a genetic algorithm. In *International Conference on Software Tools and Engineering Practice (STEP'99)*, pages 73–81, Pittsburgh, PA, 1999.

[19] Maravall D. and De Lope J. Multi-objective dynamic optimization with genetic algorithms for automatic parking. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 11:249–257, 2007.

[20] Pentico D. Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, 176(2):774–793, 2007.

[21] Shoup D. The politics and economics of parking on campus. Technical report, University of California Transportation Center, 2011.

[22] The Free Dictionary. @ONLINE. www.thefreedictionary.com. Accessed: 2012-04-24.

[23] Barata E., Cruz L., and Ferreira J. P. Parking at the uc campus: Problems and solutions. *Cities*, 2011.

[24] Carlson S. E. A general method for handling constraints in genetic algorithms. In *In Proceedings of the Second Annual Joint Conference on Information Science*, pages 663–667, 1995.

[25] Carlson S. E. Annealing a genetic algorithm over constraints. In *Proceedings of 1998 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3931–3936, 1998.

[26] Coskun M. E. Shelf space allocation: A critical review and a model with price changes and adjustable shelf heights. 2012.

[27] Goldberg D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[28] Smith A. E. and Tate D. M. Genetic optimization using a penalty function. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 499–505, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[29] Allen F. and Karjalainen R. Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, 51(2):245 – 271, 1999.

[30] Charpin J. P. F., Mbebi A., Moepya O., Kamga M., Hocking G., Ali M., and Adewumi A. Optimizing parking assignment. Technical report, Mathematics in Study Group, 2011.

[31] Djeumou F., Adewumi A., and Montaz A. Metaheuristics for space allocation problems: Comprehensive survey and review. 2010.

[32] Glover F. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5):533–549, 1986.

[33] Glover F., Laguna M., and Rafael M. Tabu search, 1997.

[34] Hoffmeister F. and Sprave J. Problem-independent handling of constraints by use of metric penalty functions. In *Evolutionary Programming*, pages 289–294, 1996.

[35] Man K. F. and Tang K. S. Genetic algorithms for control and signal processing. In *Industrial Electronics, Control and Instrumentation, 1997. IECON 97. 23rd International Conference on*, volume 4, pages 1541–1555, 1997.

[36] Vaz A. I. F. and Vicente L. N. Pswarm: a hybrid solver for linearly constrained global derivative-free optimization. *Optimization Methods Software*, 24(4-5):669–685, 2009.

[37] Brown-West O. G. Optimization model for parking in the campus environment. *Transportation Research Record: Journal of the Transportation Research Board*, 1564(1):46–53, 1996.

[38] Caron G., Hansen P., and Jaumard B. The assignment problem with seniority and job priority constraints. *Operational Research*, 47(3):449–453, 1999.

[39] Marsden G. The evidence base for parking policiesa review. *Transport Policy*, 13(6):447 – 457, 2006.

[40] Nicosia G. and Stracquadanio G. Generalized pattern search algorithm for peptide structure prediction. *Biophysical Journal*, 95(10):4988–99, 2008.

[41] Telfar G. Generally applicable heuristics for global optimisation: An investigation of algorithm performance for the euclidean traveling salesman problem. Master's thesis, Victoria University of Wellington, 1994.

[42] Terry R. G. and Soland R. M. A branch and bound algorithm for the generalized assignment problem. 3:91–103, 1975.

[43] Arsham H. Deterministic modeling: Linear optimization with applications @ONLINE. http://home.ubalt.edu/ntsbarsh/opre640a/partviii.htm. Accessed: 2013-04-16.

[44] Azami H., Mohammadi K., and Hassanpour H. An improved signal segmentation method using genetic algorithm. *International Journal of Computer Applications*, 29(8):5–9, 2011.

[45] Wolpert D. H. and Macready W. G. No free lunch theorems for search, 1995.

[46] Kazem B. I., Mahdi A. I., and Oudah A. T. Motion planning for a robot arm by using genetic algorithm. *Jordan Journal of Mechanical and Industrial Engineering*, pages 131–136, 2008.

[47] Vaz A. I. and Vicente L. N. A particle swarm pattern search method for bound constrained global optimization. *Journal of Global Optimization*, 39(2):197–219, 2007.

[48] IBM. Cplex optimizer @ONLINE. http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/. Accessed: 2013-05-30.

[49] IBM. Ibm and streetline address one of the great unsolved city problems: Parking @ONLINE. http://www-03.ibm.com/press/us/en/pressrelease/35514.wss. Accessed: 2013-04-26.

[50] IBM. Ibm global commuter pain survey: Traffic congestion down, pain way up @ONLINE. http://www-03.ibm.com/press/us/en/pressrelease/35359.wss. Accessed: 2013-04-26.

[51] Chisholm K. J. and Bradbeer P. V. G. Machine learning using a genetic algorithm to optimise a draughts program board evaluation function. In *Proceedings of IEEE International Conference on Evolutionary Computation, (ICEC'97)*, pages 715–720, 1997.

[52] Joines J. and Houck C. R. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's. In *In*, pages 579–584. IEEE Press, 1994.

[53] Millin J. J. An investigation into the use of intelligent systems for currency trading analysis. 2008.

[54] Blanton Jr., Joe L., and Wainwright R. L. Multiple vehicle routing with time and capacity constraints using genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, number 8, pages 452–459, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[55] Burke E. K. and Cowling P. Combining hybrid metaheuristics and populations for the multiobjective optimisation of space allocation problems. In *in the Proceedings of the GECCO 2001, Genetic and Evolutionary Computation Conference 2001*, pages 1252–1259. Morgan kaufmann, 2001.

[56] Fagerholt K. and Christiansen M. A travelling salesman problem with allocation, time window and precedence constraints an application to ship scheduling. *International Transactions in Operational Research*, 7(3):231–244, 2000.

[57] Goyal S. K. Two models for allocating car parking spaces. *Traffic Engineering and Control*, 19(2):83–85, 1978.

[58] Goyal S. K. and Gomes L. F. A. M. A model for allocating car parking spaces in universities. *Transportation Research Part B: Methodological*, 18(3):267–269, 1984.

[59] Hoffman K. and Padberg M. Lp-based combinatorial problem solving. *Annals of Operations Research*, 4(1):145–194, 1985.

[60] Martin L. K. Dr martin luther king @ONLINE. http://www.drmartinlutherking.net/martin-luther-king-quotes.php. Accessed: 2013-02-15.

[61] Thakur M. K., Kumari M., and Das M. Architectural layout planning using genetic algorithms. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 4, pages 5–11, 2010.

[62] Wloch K. and Bentley P. J. Optimising the performance of a formula one car using a genetic algorithm. In *In Proceedings of Eighth International Conference on Parallel Problem Solving From Nature*, pages 702–711, 2004.

[63] Applegate D. L., Bixby R. E., Chvatal V., and Cook W. J. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007.

[64] Carroll D. L. Chemical laser modeling with genetic algorithms. *AIAA Journal*, 34:338–346, 1996.

[65] Gouveia L. A classification of formulations for the (time-dependent) traveling salesman problem. *European Journal of Operational Research*, 83(1):69–82, 1995.

[66] Olsen A. L. Penalty functions and the knapsack problem. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, volume 2, pages 554–558, 1994.

[67] Winston W. L. and Goldberg J. B. *Operations research: applications and algorithms*. Thomson Brooks & Cole, 2004.

[68] Amini M. M. and Racer M. A hybrid heuristic for the generalized assignment problem. *European Journal of Operational Research*, 87(2):343–348, 1995.

[69] Chiu H. M. A location model for the allocation of the off-street parking facilities. *Journal of the Eastern Asia Society for Transportation Studies*, 6:1344–1353, 2005.

[70] Gen M. and Cheng R. A survey of penalty techniques in genetic algorithms. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 804–809, 1996.

[71] Gen M. and Cheng R. *Genetic Algorithms and Engineering Optimization.* Wiley Series in Engineering Design and Automation. John Wiley & Sons, 1999.

[72] Glavic M. and Wehenkel L. Interior point methods: A survey, short survey of applications to power systems, and research opportunities. Technical report, Technical Report, 2004.

[73] Lewis R. M. and Torczon V. Pattern search algorithms for bound constrained minimization. Technical report, ICASE, NASA Langley Research, 1996.

[74] Lewis R. M., Torczon V., and Trosset M. W. Direct search methods: Then and now. *Journal of Computational and Applied Mathematics*, 124:191–207, 2000.

[75] Salkin H. M. and De Kluyver C. A. *The Knapsack Problem: A Survey.* Reprint / Purdue university, Krannert Graduate School of Management. Krannert Graduate School of Management and Institute for Research in the Behavioral, Economic, and Management Sciences, 1980.

[76] Silvano M. and Paolo T. *Knapsack problems: algorithms and computer implementations.* John Wiley & Sons, Inc., New York, NY, USA, 1990.

[77] Spronck P. H. M. and Kerckhoffs E. J. H. Using genetic algorithms to design neural reinforcement controllers for simulated plants. In *Proceedings of the 11th European Simulation Conference*, pages 292–299, 1997.

[78] MATLAB. *version 7.10.0 (R2010a).* The MathWorks Inc., Natick, Massachusetts, 2010.

[79] Gabere M. N. Simulated annealing driven pattern search algorithms for global optimization. Master's thesis, University of the Witwatersrand, 2007.

[80] Sivanandam S. N. and Deepa S. N. *Introduction to Genetic Algorithms.* Springer Publishing Company, Incorporated, 1st edition, 2007.

[81] Parking Network. About parking @ONLINE. http://www.parking-net.com/about-parking. Accessed: 2013-04-10.

[82] Adewumi A. O. *Some Improved Genetic-algorithms Based Heuristics for Global Optimization with Innovative Applications.* PhD thesis, University of the Witwatersrand, 2010.

[83] Yeniay O. Penalty function methods for constrained optimization with genetic algorithms. *Mathematical and Computational Applications*, 10:45–56, 2005.

[84] University of KwaZulu-Natal. *New Policy: Staff Parking Fees and Reserved Bays.* Durban, South Africa, 2006.

[85] Alberto P., Nogueira F., Rocha H., and Vicente L. Pattern search methods for user-provided points: Application to molecular geometry problems. *SIAM Journal on Optimization*, 14(4):1216–1236, 2004.

[86] Chan P. and Lyu M. R. Digital video watermarking with a genetic algorithm. In *Proceedings of International Conference on Digital Archive Technologies (ICDAT 05)*, pages 139–153, 2005.

[87] Sattayhatewa P. and Smith R. L. Development of parking choice models for special events. *Transportation Research Record: Journal of the Transportation Research Board*, 1858:31–38, 2003.

[88] Shabanzadeh P., Hassan M. A., June L. W., and Mohagheghtabar M. Using pattern search methods for minimizing clustering problems. *World Academy of Science, Engineering & Technology*, 62:158 – 162, 2010.

[89] Venkataraman P. *Applied Optimization with MATLAB Programming.* Wiley Publishing, 2nd edition, 2009.

[90] Garcia E. R. Q. and Quintero G. C. M. Space allocation using intelligent optimization techniques. In *ANDESCON, 2010 IEEE*, pages 1–6. IEEE, 2010.

[91] Hooke R. and Jeeves T. A. " direct search" solution of numerical and statistical problems. *Journal of the ACM*, 8(2):212–229, 1961.

[92] Lopes R. and Girimonte D. The office-space-allocation problem in strongly hierarchized organizations. In *Proceedings of the 10th European conference on Evolutionary Computation in Combinatorial Optimization*, pages 143–153, Berlin, Heidelberg, 2010. Springer-Verlag.

[93] Pereira R., Cummiskey K., and Kincaid R. Office space allocation optimization. In *Systems and Information Engineering Design Symposium (SIEDS), 2010 IEEE*, pages 112–117, 2010.

[94] Reeves C. R., editor. *Modern heuristic techniques for combinatorial problems.* John Wiley & Sons, Inc., New York, NY, USA, 1993.

[95] Eberhart R.C. and Shi Y. Particle swarm optimization: developments, applications and resources. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 81–86, 2001.

[96] Alsumait J. S., Sykulski J. K., and Alothman A. K. Application of pattern search method to power system economic load dispatch. In *Third IASTED Asian Conference Power and Energy Systems*, pages 90–95, 2007.

[97] Arora S. and Puri M. C. A variant of time minimizing assignment problem. *European Journal of Operational Research*, 110(2):314 – 325, 1998.

[98] Cernic S., Jezierski E., Britos P., Rossi B., and García M. R. Genetic algorithms applied to robot navigation controller optimization. In *International Conference on Intelligent Systems and Control*, pages 230–234, 1999.

[99] Geisendorf S. Genetic algorithms in resource economic models. *Santa Fe Institute, NM, USA, Working Papers*, pages 99–08, 1999.

[100] Geisendorf S. Are genetic algorithms a good basis for economic learning models? Technical Report 5, University of Kassel, Institute of Economics, 2007.

[101] Helvig C. S., Robins G., and Zelikovsky A. The moving-target traveling salesman problem, 2003.

[102] Kazarlis S. and Petridis V. Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms. In *Parallel Problem Solving from Nature V—PPSN V*, pages 211–220. Springer-Verlag, 1998.

[103] Kirkpatrick S., Gelatt C. D., and Vecchi M. P. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[104] Mizuta S., Sato T., Lao D., and Ikeda T. M. Structure design of neural networks using genetic algorithms. In *Complex Systems*, volume 13, pages 161–175, 2001.

[105] Nemirovski A. S. and Todd M. J. Interior-point methods for optimization. *Acta Numerica*, 17(1):191–234, 2008.

[106] White M. S. and Flockton S. J. Genetic algorithms for digital signal processing. In *Lecture Notes in Computer Science*, 1994.

[107] Zekai S. and Oztopal A. Genetic algorithm for the classification and prediction of precipitation occurrence. *Hydrological Sciences-journal-des Sciences Hydrologiques*, 46(2):255–267, 2001.

[108] Hegazy T. Optimization of resource allocation and leveling using genetic algorithms. *Journal of Construction Engineering & Management*, 125(3):167, 1999.

[109] Litman T. *Parking management: strategies, evaluation and planning.* Victoria Transport Policy Institute, 2011.

[110] Weise T. Global optimization algorithms theory and application , 2008.

[111] Yokota T., Gen M., Ida K., and Taguchi T. Optimal design of system reliability by an improved genetic algorithm. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 79(2):41–51, 1996.

[112] Aickelin U. *Genetic algorithms for multiple-choice optimisation problems*. PhD thesis, University of Swansea, 1999.

[113] Torczon V. On the convergence of pattern search algorithms. *Society for Industrial and Applied Mathematics Journal on Optimization*, 7(1):1–25, 1997.

[114] Chinneck J. W. Practical optimization: A gentle introduction @ONLINE. http://www.sce.carleton.ca/faculty/chinneck/po.html. Accessed: 2013-05-09.

[115] Coit D. W. and Smith A. E. Penalty guided genetic search for reliability design optimization. *Computers and Industrial Engineering*, 30:895–904, 1996.

[116] Glover F. W. and Kochenberger G. A., editors. *Handbook of Metaheuristics*, volume 114 of *International Series in Operations Research & Management Science*. Springer, 2003.

[117] Sun W., Heights U., Mouskos K. C., and Bernstein D. A web-based parking reservation system. 2003.

[118] Young W., Thompson R. G., and Taylor M. A. P. A review of urban car parking models. *Transport Reviews*, 11:63–84, 1991.

[119] Chu W-M., Yao M-J., and Tseng T-Y. An improved genetic algorithm for solving the optimal resource allocation problem in stochastic activity networks. In *Proceedings of the Fifth Asia Pacific Industrial Engineering and Management Systems Conference*, 2004.

[120] Wikipedia. Branch and bound @ONLINE. https://en.wikipedia.org/wiki/Branch-and-bound. Accessed: 2013-06-28.

[121] Wikipedia. Cplex @ONLINE. http://en.wikipedia.org/wiki/CPLEX. Accessed: 2012-08-26.

[122] Wikipedia. Metaheuristic @ONLINE. http://en.wikipedia.org/wiki/Metaheuristic. Accessed: 2012-05-15.

[123] Wikipedia. Parallel parking @ONLINE. http://en.wikipedia.org/wiki/Parallel-parking. Accessed: 2011-11-24.

[124] Wikipedia. Parking lot @ONLINE. http://en.wikipedia.org/wiki/Parking-lot. Accessed: 2011-11-24.

[125] Wikipedia. Simplex algorithm @ONLINE. http://en.wikipedia.org/wiki/Simplex-algorithm. Accessed: 2013-06-28.

[126] Hu X. Particle swarm optimization @ONLINE. http://www.swarmintelligence.org/index.php. Accessed: 2013-04-05.

[127] Hu X., Shi Y., and Eberhart R. Recent advances in particle swarm. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 1, pages 90–97, 2004.

[128] Song X., Li D., Gu H., Liao Y., and Ren D. Insights into performance of pattern search algorithms for high-frequency surface wave analysis. *Computers & Geosciences*, 35(8):1603–1619, 2009.

[129] Xin-She Y. Engineering optimization an introduction with metaheuristic applications, 2010.

[130] Xin-She Y. Review of metaheuristics and generalised evolutionary walk algorithm. *International Journal of Bio-Inspired Computation*, 3(2):77–84, 2011.

[131] Ahmed Z. and Majeed S. Machine learning and data optimization using bpnn and ga in doc. *International Journal of Emerging Sciences*, 1(2):108–119, 2011.

[132] Michalewicz Z. Genetic algorithms, numerical optimization, and constraints. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, volume 195, pages 151–158, 1995.

[133] Michalewicz Z. and Attia N. F. Evolutionary optimization of constrained problems. In *Proceedings of the 3rd annual conference on evolutionary programming*, pages 98–108, 1994.

[134] Michalewicz Z. and Nazhiyath G. Genocop iii: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In *Evolutionary Computation, 1995., IEEE International Conference*, volume 2, pages 647–651, 1995.

[135] Michalewicz Z. and Schoenauer M. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[136] Wang Z. and Zhou W. Current situation and improvement strategy for campus parking in china. In *Proceedings of the 2010 International Conference on Intelligent Computation Technology and Automation*, volume 01, pages 1075–1078, 2010.

Appendix A

Dataset 1

Table A.1: Available parking spaces in the parking lots

Parking lots	Reserved	Unreserved	Total space
1	25	47	72
2	65	121	186
3	39	71	110
4	69	129	198
5	52	95	147
6	36	68	104
7	56	104	160
8	49	92	141
Total	391	727	1118

Table A.2: Population of users to be allocated parking

Buildings	Reserved users	Unreserved users	Total users
1	27	65	92
2	9	28	37
3	27	66	93
4	51	102	153
5	50	100	150
6	43	89	132
7	25	62	87
8	30	70	100
9	28	67	95
10	33	75	108
11	29	68	97
12	39	84	123
Total	391	876	1267

Table A.3: The distance cost

Building	Parking Lots							
	1	2	3	4	5	6	7	8
1	79	160	26	143	183	93	141	195
2	108	86	196	62	146	170	43	147
3	182	100	56	83	152	15	176	125
4	57	170	125	156	191	15	183	59
5	143	128	85	13	80	48	11	152
6	97	187	24	147	97	151	26	69
7	183	55	67	157	184	145	78	155
8	48	116	159	93	161	85	43	163
9	168	15	80	66	77	126	11	124
10	94	53	83	169	81	58	109	93
11	189	45	26	106	143	63	60	55
12	136	95	123	196	125	58	61	169

Table A.4: Parking permit issued for each parking lot

Parking lots(k)	Parking permit issued(A_k)
1	75
2	217
3	121
4	232
5	167
6	113
7	183
8	159
Total	1267

Appendix B

Dataset 2

Table B.1: Available parking spaces in the parking lots

Parking lots	Reserved	Unreserved	Total space
1	20	37	57
2	27	51	78
3	27	51	78
4	48	90	138
5	33	61	94
6	44	82	126
7	39	71	110
8	45	83	128
9	21	40	61
10	52	96	148
Total	356	662	1018

Table B.2: Population of Users to be Allocated parking

Buildings	Reserved users	Unreserved users	Total users
1	31	72	103
2	17	51	68
3	45	117	162
4	32	73	105
5	8	32	40
6	39	89	128
7	29	72	101
8	7	35	42
9	41	111	152
10	21	60	81
11	15	48	63
12	9	39	48
13	20	55	75
14	15	48	63
15	27	87	114
Total	356	989	1345

Table B.3: The Distance Cost

Building	Parking lots									
	1	2	3	4	5	6	7	8	9	10
1	88	78	56	51	167	138	84	93	143	48
2	10	114	35	86	119	57	92	159	81	85
3	113	76	53	73	161	100	192	38	195	199
4	49	128	77	54	73	86	119	128	195	86
5	52	161	65	188	52	124	171	60	132	135
6	72	152	186	140	69	162	63	95	173	181
7	28	34	20	193	121	30	128	170	86	199
8	152	166	123	93	168	166	122	47	130	134
9	152	15	41	189	65	170	193	68	197	31
10	113	89	169	11	86	77	26	102	116	17
11	74	149	42	126	174	92	105	74	187	127
12	168	158	105	162	127	119	109	162	147	118
13	115	80	200	54	198	143	27	198	102	193
14	192	152	78	187	49	151	182	40	131	152
15	180	180	19	155	167	154	178	55	179	136

Table B.4: Parking permit issued for each parking lot

Parking lots(k)	Parking permit issued(A_k)
1	73
2	102
3	102
4	184
5	124
6	168
7	146
8	170
9	78
10	198
Total	1345

Appendix C

Dataset 3

Table C.1: Available Parking Spaces in the Parking Lots

Parking lots	Reserved	Unreserved	Total space
1	11	21	32
2	59	110	169
3	38	71	109
4	19	36	55
5	52	97	149
6	43	79	122
7	50	94	144
8	79	146	225
9	22	41	63
10	44	82	126
11	15	28	43
12	40	74	114
13	66	123	189
14	18	33	51
15	62	115	177
16	33	62	95
17	57	105	162
Total	708	1317	2025

Table C.2: Population of users to be allocated parking

Buildings	Reserved users	Unreserved users	Total users
1	78	184	262
2	110	231	341
3	42	130	172
4	25	102	127
5	20	91	111
6	67	165	232
7	28	106	134
8	35	118	153
9	13	57	70
10	71	173	244
11	119	250	369
12	100	226	326
Total	708	1833	2541

Table C.3: The distance cost

Building	Parking lots																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	247	192	77	163	284	165	237	103	85	90	246	120	163	274	167	220	243
2	292	97	233	252	215	137	93	259	242	88	182	250	253	186	165	79	275
3	95	131	293	248	149	130	79	265	292	84	193	249	282	237	252	109	85
4	283	229	202	121	115	165	94	181	147	227	156	289	218	81	256	122	249
5	61	188	230	67	262	109	207	169	298	166	230	161	143	163	98	93	97
6	110	86	126	64	286	57	260	272	132	78	68	164	151	69	56	131	57
7	52	145	165	209	144	215	178	66	84	225	199	200	160	216	64	250	82
8	218	149	62	156	67	90	91	177	146	95	265	261	220	226	86	125	83
9	276	194	146	276	95	251	229	205	191	251	162	58	166	280	93	244	82
10	193	55	140	154	194	152	277	233	208	178	213	97	288	215	206	188	284
11	89	194	122	89	96	132	105	107	185	187	126	286	139	223	57	189	118
12	176	283	254	185	123	237	268	55	129	102	202	287	135	263	168	233	286

Table C.4: Parking permit issued for each parking lot

Parking lots(k)	Parking permit issued(A_k)
1	35
2	216
3	135
4	65
5	189
6	153
7	182
8	291
9	75
10	158
11	49
12	142
13	243
14	59
15	226
16	117
17	206
Total	2541

Appendix D

Dataset 4

Table D.1: Available parking spaces in the Parking lots

Parking lots	Reserved	Unreserved	Total space
1	37	69	106
2	27	49	76
3	41	77	118
4	26	47	73
5	34	62	96
6	31	57	88
7	36	68	104
8	52	97	149
9	44	82	126
10	52	96	148
11	26	47	73
12	36	67	103
13	19	36	55
14	44	82	126
15	39	71	110
16	48	88	136
17	52	97	149
18	50	93	143
Total	694	1285	1979

Table D.2: Population of users to be allocated parking

Buildings	Reserved users	Unreserved users	Total users
1	14	40	54
2	15	42	57
3	35	78	113
4	17	49	66
5	21	56	77
6	25	61	86
7	25	63	88
8	24	61	85
9	36	79	115
10	12	41	53
11	14	45	59
12	11	34	45
13	21	56	77
14	45	92	137
15	16	49	65
16	43	89	132
17	40	85	125
18	45	94	139
19	49	98	147
20	19	53	72
21	20	55	75
22	31	72	103
23	21	56	77
24	48	97	145
25	47	95	142
Total	694	1640	2334

Table D.3: The distance cost

Building	Parking lots																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	335	315	147	172	107	354	196	109	276	393	387	366	71	190	156	92	198	148
2	367	310	288	119	328	80	67	277	183	204	241	362	289	234	295	256	83	204
3	94	187	279	138	159	190	366	306	334	89	232	167	65	196	283	129	259	234
4	370	279	107	266	235	141	381	277	236	140	131	295	75	280	239	185	215	210
5	271	110	92	216	108	330	222	208	173	193	221	119	233	270	294	254	294	356
6	84	297	224	173	261	201	221	241	379	258	268	61	84	152	283	138	295	231
7	147	61	386	341	142	369	168	154	357	142	288	310	336	201	112	152	273	380
8	241	147	169	255	279	114	365	311	243	261	188	225	336	55	95	266	62	273
9	385	66	255	242	291	142	179	116	268	299	179	218	303	394	400	143	74	385
10	388	84	128	371	312	101	89	290	255	128	396	367	102	109	110	339	162	134
11	105	338	313	150	208	98	323	114	123	91	63	263	281	87	61	394	236	287
12	390	293	139	315	79	354	186	179	155	154	360	266	232	180	246	306	279	151
13	385	161	227	314	130	253	135	269	215	162	370	351	391	119	359	170	193	285
14	220	383	295	183	370	242	191	323	131	198	329	332	277	221	284	254	337	293
15	330	62	362	249	103	101	84	78	346	228	85	252	330	169	117	88	301	74
16	100	204	386	77	339	349	96	375	118	80	142	114	209	383	179	367	389	139
17	198	184	242	69	238	268	380	321	129	142	167	134	201	372	211	358	236	128
18	371	318	99	236	399	173	385	220	110	330	288	360	339	68	394	336	164	284
19	327	328	102	323	77	230	251	203	130	60	98	60	79	308	105	141	87	346
20	386	115	140	377	205	191	71	206	202	375	302	221	97	144	349	258	264	171
21	280	221	344	95	87	77	132	157	159	306	87	109	111	198	276	58	323	323
22	62	206	139	249	387	134	174	228	373	221	279	393	187	242	182	199	198	286
23	347	276	335	214	52	93	337	229	201	252	223	299	341	380	117	159	82	52
24	377	298	135	54	321	114	55	336	115	133	323	225	331	196	200	107	143	261
25	288	314	375	168	336	134	65	328	367	211	300	215	71	394	219	113	104	185

Table D.4: Parking permit issued for each parking lot

Parking lots(k)	Parking permit issued(A_k)
1	124
2	89
3	140
4	82
5	111
6	101
7	122
8	180
9	150
10	179
11	82
12	120
13	60
14	150
15	129
16	163
17	180
18	172
Total	2334

Appendix E

Conference paper accepted at iSTEAMS Research Nexus 2013

An Exact Solution for Allocating Car Parking Spaces on Campus

Luke O. Joel[*], Sawyerr A. Babatunde[**] and Adewumi O. Aderemi[*]

[*]School of Mathematics, Statistics and Computer Science, University of Kwazulu-Natal, Westville Campus, Durban, South Africa
[**]Department of Computer Sciences, University of Lagos, Yaba, Lagos, Nigeria

Abstract

All over the world, especially in the university environment, planning managers and traffic engineers are constantly faced with the problem of inadequate allocation of car parking spaces to demanded users. Users could either prefer reserved parking spaces to unreserved parking spaces or vice versa. This makes the campus parking manager to be faced with two basic problem which are: the problem of allocating the actual number of available reserved spaces to users without any conflict over the same parking space, and the problem of determining the number of parking permit to be issued for parking lot with unreserved spaces. Hence, an optimal or available solution to the problem is required. This paper investigates a model for allocating car parking spaces, adds a constraint to address the reserved parking policy in a university environment and solves the parking allocation problem using an exact solution method. The result obtained gives the value of the objective function and the optimal allocation of users to each parking lot.

Keywords: Allocation, Model, Parking space, Parking lot, Reserved spaces, University

1 Introduction

Parking is a major concern in the transportation planning and traffic management of any organisation all over the world. Parking problems, among other things, are major problems facing the society and especially the university environment due to limited number of available parking spaces and the cost of parking facilities. The challenge is to develop a model of the problem that considers different parking policies in the campus environment and to obtain an optimal or available solution to the problem. Many studies have looked at the

parking problem from administrative and management point of view. However this paper will examine the problem from optimization point of view. The paper addresses the problem of parking allocation in the university environment by formulating a model of the problem which caters for both reserved and unreserved policy in the campus world

2 Related Works

In the previous studies on parking problem, Narragon, Dessouky and DeVor [4] evaluated campus parking over-issuance policies by developing a probabilistic model which permits different classes of users to be considered simultaneously. Mouskos et al. [3] formulated a deterministic dynamic parking reservation system (PRS) for performing parking space assignment on the minimization of parking cost in order to aid users in securing a parking space either before or during their trip. Chiu [5] developed a multi-objective linear integer programming model for the optimum allocation of the off-street parking facilities decision makers. He advocated for the use of existing public facility as a parking facility. Batabyal et al. [7] analysed two university parking issues by determining the mean parking time of an arriving car for both short term and long term parkers and computing their probability distribution function. He also calculated the probability distribution function of parking violators. Sattayhatewa et al. [8] modelled the evaluation of parking lot choice by considering three (3) major factors- driving time, parking cost, and walking time which could be used to analyse the current traffic conditions, improve the traffic conditions and assess various operational and management policies for special events. Brown-West [9] presented an optimization methodology for the use of existing land and to manage parking spaces in a competitive, policy-driven university campus. Major operational and site features, as well as parameters that could help parking managers and engineers are included in the model. Essentially, Goyal and Gomes [1] proposed a parking allocation model in a university environment on cases where the number of users is equal or less than the available spaces and where the number of users is greater than the available parking spaces. The latter case will be the focus of this paper with reserved policy.

3 Parking Allocation Model

The allocation of available parking spaces to a set of users in order to minimize the distance walked by each user from the parking lot to the buildings in which they work is a difficult one, especially when the reserved policy is to be considered. A parking reserved policy is an important part of campus parking, hence there is need to incorporate this into the campus parking space allocation model. A constraint that addresses the reserved policy, Equation 1, was introduced to the model proposed in [1] for a case where the number of users, T_U,

is greater than the available spaces, T_S. The constraint ensure that sum of the parking allocation, X_{ijk}, is equal to the number of available spaces, M_{ik}, for reserved policy and greater than the number of available spaces for unreserved policy.

$$\sum_{j=1}^{m} X_{ijk} \geq M_{ik} \quad \text{for } i = 1, 2, ..., l \text{ and } k = 1, 2, ..., n \tag{1}$$

By reserved spaces, we mean, a user given a reserved allocation does not share his allocation with any other user. That is, the number of users allocated to a parking space marked reserved cannot be more than the number of parking spaces. For unreserved, the number of users could be more than the available parking spaces because they are meant to be shared by more than one user. Hence, the model is formulated as an linear programming model and it is given as:

$$Minimize \quad Z = \sum_{k=1}^{n} \sum_{j=1}^{m} \sum_{i=1}^{l} D_{jk} X_{ijk} \tag{2}$$

subject to:

$$\sum_{k=1}^{n} X_{ijk} = P_{ij} \quad \text{for } i = 1, 2, ..., l \text{ and } j = 1, 2, ..., m \tag{3}$$

$$\sum_{j=1}^{m} \sum_{i=1}^{l} X_{ijk} = A_k \quad \text{for } k = 1, 2, ..., n \tag{4}$$

$$\sum_{j=1}^{m} X_{ijk} \geq M_{ik} \quad \text{for } i = 1, 2, ..., l \text{ and } k = 1, 2, ..., n \tag{5}$$

$$X_{ijk} \geq 0 \qquad \forall \quad i, j, k \geq 0 \tag{6}$$

Where,
l = the total number of permits type (with index i)
m = the total number of users' building (with index j)
n = the total number of parking lot (with index k)

$$T_S = \sum_{k=1}^{n} N_k = \sum_{k=1}^{n} \sum_{i=1}^{l} M_{ik}$$

T_S = the total number of available parking spaces
N_k = the number of available spaces in the kth parking lot excluding the spaces for handicapped users

M_{ik} = the number of parking places available with permit type i in kth parking lot
A_k = the number of permits issued to the kth parking lot
D_{jk} = the distance between the jth users' building and the kth parking lot
X_{ijk} = the number of people having permit type i, users' building j in the kth parking lot

$$T_U = \sum_{i=1}^{l} B_i = \sum_{j=1}^{m} \sum_{i=1}^{l} P_{ij}$$

T_U = the total number of users demanding parking
B_i = the number of permit type i users
P_{ij} = the number of permit type i users working in building j

The objective function in Equation 2 minimizes the distances walked by users from each parking lot to their respective buildings. Equation 3 is the permit type users constraint which ensures that the sum of the parking allocation in each parking lot is equal to the number of users with the permit type for the parking lot. Since several parking permits are issued for different parking lot, the constraint in Equation 4 ensures that the sum of parking allocation for users with permit type i working in building j is equal to the parking permit issued for the parking lot. Equation 5 is the reserved spaces constraint introduced to the model and it is as explained earlier. The non-negativity constraint in Equation 6 keeps the variables to be equal or greater than zero.
[1] made the following assumptions:

1. The shortest walking distance between each parking lot and the users' working building is known and it is taken by all users.

2. The probability of a user bringing his car on a particular day and the probability of the user finding a space on that day is the same for all users.

4 Data

The data used is from the parking data for University of KwaZulu-Natal (UKZN), Westville Campus [6]. UKZN has a staff population of approximately 4300 people, and about 40% of this were from Westville Campus, which is approximately 1720 people. Obviously, not all the 1720 people will need a parking space, so about 75% of the Westville Campus staff require parking spaces, which gives us a population of 1290 users demanding parking spaces. A look at the available parking spaces is necessary for the efficient calculation of the ratio of demand to supply of parking spaces on Westville campus. Hence, from Table 1, we are

to allocate 1047 parking spaces to 1290 users with some reserved consideration. There are several buildings and parking lots in Westville Campus but twelve(12) out of these buildings and six(6) out of these parking lots are used in the study. A break down of the users demanding parking in each of this building is given in Table 2. The distance cost from each building to each parking lot is calculated and given in Table 3.

Table 1: Available Parking Spaces in the Parking Lots

Parking Lots	Number Available	Reserved	Unreserved
1	201	40	161
2	138	138	-
3	126	27	99
4	142	32	110
5	68	68	-
6	372	72	300
Total	1047	377	670

Table 2: Population of Users demanding parking

Buildings	Users demanding parking	Reserved Number	Unreserved Number
1	142	41	101
2	77	23	54
3	118	34	84
4	64	19	45
5	60	19	41
6	220	64	156
7	51	15	36
8	129	38	91
9	42	11	31
10	103	30	73
11	169	49	120
12	115	34	81
Total	1290	377	913

The mean and the standard deviation [1] for the distribution is given as $p.A_k$ and $\sqrt{(p(1-p)A_k)}$, where p is the probability of a user bringing his car on a particular day. Goyal & Gomes [1] observed that in order to get equal probability for all the users, the

Table 3: The Distance Cost

Building	Parking Lots					
	1	2	3	4	5	6
1	255	270	440	165	285	610
2	150	165	335	60	180	505
3	165	180	320	75	195	490
4	120	135	275	120	150	445
5	270	285	260	105	200	430
6	180	150	215	195	60	485
7	60	90	320	150	180	490
8	90	60	290	165	150	400
9	350	320	210	260	245	90
10	440	410	120	350	335	180
11	320	290	60	230	215	200
12	335	305	75	245	230	215

equation in (7) must be satisfied

$$\frac{N_k - p.A_k}{\sqrt{(p(1-p)A_k)}} = \Psi \tag{7}$$

Getting the value of Ψ enables us to calculate the total number of permit issued for the kth parking lot. However, In order to obtain the value of Ψ that will be used to calculate the total number of permit issued, A_k, which will be equal to the number of users demanding parking, T_U, Goyal [2] suggested squaring Equation (7), rearranging the terms for the kth parking lot, ignoring small terms and then equating it to the total number of users demanding parking. The resultant equation is the quadratic equation in (8).

$$n\Psi^2 - 2\Psi \sum_{k=1}^{n} \sqrt{N_k} - 2(p.T_U - \sum_{k=1}^{n} N_k) = 0 \tag{8}$$

Solving the quadratic equation in (8), the value of Ψ obtained is used to get the values of A_k in equation (9):

$$A_k = \frac{(2N_k + \Psi^2) - 2\Psi\sqrt{(N_k)}}{2p} \tag{9}$$

Parking Permit Calculation

The numbers of parking permit to be issued for each parking lot are calculated. These numbers are only calculated for the parking lots that are not entirely for reserved parking,

since the numbers of parking permit issued cannot be greater than the numbers of available parking spaces in a reserved parking. Hence, the number of parking permit issued will not be calculated for parking lot 2 and 5. We subtract the number of parking spaces in the two parking lots for reserved parking lots only - 206, from the total number of users demanding parking - 1290, to obtain the number of users - 1084 to be used in calculating the parking permit issued. The values of the variables to be substituted into Equation(8) are

$$B = 1290, N_1 = 201, N_2 = 138, N_3 = 126,$$
$$N_4 = 142, N_5 = 68, N_6 = 372, n = 4,$$

with $p = 0.7$ as suggested by [2, 4].
which gives

$$4\Psi^2 - 2\Psi(\sqrt{N_1} + \sqrt{N_3} + \sqrt{N_4} + \sqrt{N_6})$$
$$-2(0.7 * 1084 - (N_1 + N_3 + N_4 + N_6)) = 0$$

and finally the quadratic equation

$$6\Psi^2 - 113.2122\Psi + 164.4 = 0$$

Solving the quadratic equation, we obtain the practical value of $\Psi = 1.535$

Hence, we put the value of Ψ got into Equation (9) to get the parking permit issued for each parking lot. Table 4 gives the detail of that. Parking Lots 2 and 5 are only for reserved parking, while the other parking lots have a certain number for reserved and unreserved parking as given in Table 1 .

Table 4: Parking Permit Issued for each Parking Lot

Parking Lots(k)	Number Available(N_k)	Parking Permit Issued(A_k)
1	201	258
$*2$	138	138
3	126	157
4	142	178
$*5$	68	68
6	372	491
Total	1047	1290

5 Results and Discussion

The mathematical model was implemented using IBM ILOG CPLEX software. The IBM ILOG CPLEX optimization studio (simply called CPLEX version 12.4) uses a variants of simplex method or the barrier interior point method to solve different kind of optimization problems. The CPLEX software package is incorporated into the Advanced Interactive Multidimensional Modeling System (AIMMS) which is used to obtain optimal solution to the problem. The following is the discussion of the results obtained by using the CPLEX software package.

Figure 1 gives the parking allocation for the formulated model with reserved constraint while Figure 2 gives the parking allocation for the model formulated without reserved constraint in [1]. Reserved space is abbreviated to 'Rv' and Unreserved space to 'UnRv' in the results shown in Figure 1, 2, 3, and 4. The value of the objective function, Z, in Equation (2), is the minimized value of the distances walked by the users from each parking lot to their respective buildings provided the constraints are satisfied. The objective value obtained for the allocation in Figure 1 is 229160 while that of the allocation in Figure 2 is 210395. Although the objective value of the allocation in Figure 2 gives a minima value of both but the allocation obtained is infeasible based on the data. Comparing the two allocations, Figure 1 indicates that the reserved parking spaces were allocated to reserved users. But in Figure 2, the allocation is contrary. That is, the number of users that are assigned to reserved parking in Figure 2 are more than the number of reserved spaces available. Parking lot 2 and 5 are for reserved users, see Table 1 and 4. This was emphasized by the parking allocation shown in Figure 1 but not in Figure 2. Also, the number of spaces for reserved parking in parking lot 1,3,4, and 6 are the same with the number of allocated reserved users for these parking lots only in Figure 1.

Figure 3 gives a close comparison of the total number of allocated spaces in each parking lot for the two models. Figure 4 is a graph representing the same values as shown in Figure 3. The number of reserved and unreserved spaces allocated using the model with reserved constraints is highlighted in yellow. the results indicate that adding a constraint to address the reserved policy in a campus environment to the model formulated in [1] is necessary in order to obtain feasible solution to the campus parking space allocation problem. In general, the allocation that was done assigned as much as possible the closest parking lot to users in a particular building while considering the interest of the remaining users in other building. Where the closest parking lot would not be possible, a little farther parking lot would be considered.

	Parking Lot 1		Parking Lot 2		Parking Lot 3		Parking Lot 4		Parking Lot 5		Parking Lot 6	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
Building 1	40	93	1					8				
Building 2			10				13	54				
Building 3		45	34					84				
Building 4			19									
Building 5		36					19					41
Building 6		44				130			64			26
Building 7			15									
Building 8			38									47
Building 9											11	31
Building 10											30	73
Building 11			21								28	120
Building 12					27				4		3	81
Total Allocation	40	218	138	-	27	130	32	146	68	-	72	419

Figure 1: Parking Allocation with Reserved Constraint

	Parking Lot 1		Parking Lot 2		Parking Lot 3		Parking Lot 4		Parking Lot 5		Parking Lot 6	
	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv	Rv	UnRv
Building 1	41	101										
Building 2	10						13	54				
Building 3					5		27	84			2	
Building 4	19	36		9								
Building 5											19	41
Building 6					64	88				68		
Building 7	15	36										
Building 8			38	91								
Building 9											11	31
Building 10											30	73
Building 11											49	120
Building 12					27						34	81
Total Allocation	85	173	38	100	69	88	40	138	-	68	145	346

Figure 2: Parking Allocation without Reserved Constraint

	Data Given		Model proposed		Model in [1]	
	Rv	UnRv	Rv	UnRv	Rv	UnRv
Parking lot 1	40	161	40	218	85	173
Parking lot 2	138	-	138	-	38	100
Parking lot 3	27	99	27	130	69	88
Parking lot 4	32	110	32	146	40	138
Parking lot 5	68	-	68	-	-	68
Parking lot 6	72	300	72	419	145	346

Figure 3: Comparing the Allocation with the Data given

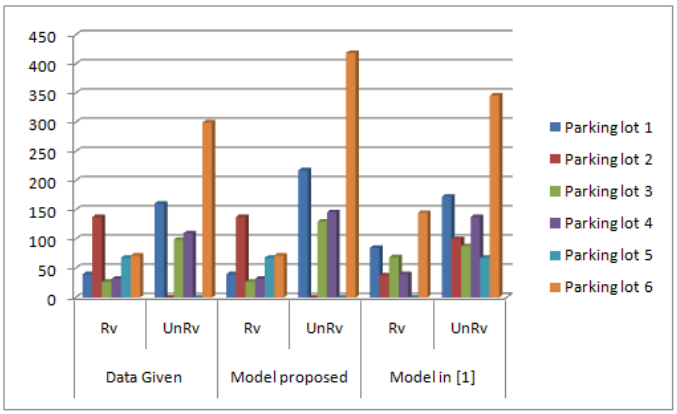

Figure 4: Comparing the Allocation with the Data given 2

6 Conclusion

The model for allocating car parking spaces in the university with reserved constraint policy was investigated. An added constraint was introduced to the model proposed in [1] so as to accommodate the reserved policy which is an important part of any university transportation planning. Some parking data were use to test the model. An exact solution, the optimum objective function value and the allocation of users to each parking lot were obtained using the CPLEX software.

References

[1] Goyal, S. K. and Gomes, L. F. A. M. (1984), A model for allocating car parking spaces in universities, Transportation Research Part B: Methodological, volume 18, number 3, pages 267-269.

[2] Goyal S. K. (1978), Two models for allocating car parking spaces, Traffic engineering and control, volume 19, number 2, pages 83-85.

[3] Mouskos, K.C. and Tvantzis, J. and Bernstein, D. and Sansil, A.(2000), Mathematical formulation of a deterministic parking reservation system (PRS) with fixed costs, Electrotechnical Conference,MELECON 2000,volume 2, pages 648-651.

[4] Narragon E. A., Dessouky M. I. and DeVor R. E. (1974), A Probabilistic Model for Analyzing Campus Parking Policies, Operations Research, volume 22, number 5, pages 1025-1039.

[5] Hsien-Ming C. (2005), A Location Model for the Allocation of the off-street Parking Facilities, Eastern Asia Society for Transportation Studies, volume 6, pages 1344-1353.

[6] University of KwaZulu-Natal Parking Policy (2006), Durban, South Africa

[7] Batabyal A. A. and Nijkamp P. (2010), A probabilistic analysis of two university parking issues, The Annals of Regional Science, volume 44, number 1, pages 111-120.

[8] Sattayhatewa P. and Smith R. L. (2003), Development of parking choice models for special events, Transportation Research Record: Journal of the Transportation Research Board, volume 1858, pages 31-38.

[9] Brown-West O. G. (1996), Optimization model for parking in the campus environment, Journal of the Transportation Research Board, volume 1564, number 1, pages 46-53.