Studies of Heuristics for Hostel Space Allocation Problem

Ariyo Sunday Ajibola

A thesis submitted in fulfilment of the requirements
for the degree of Masters of Science in Computer Science

in the

School of Mathematic, Statistics and Computer Science
University of KwaZulu-Natal
Durban, South Africa

July 2013

Examiner's Copy

UNIVERSITY OF KWAZULU-NATAL

COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE

DECLARATION

The research described in this thesis was performed at the University of KwaZulu-Natal under the supervision of Dr. A. O. Adewumi. I hereby declare that all materials incorporated in this thesis are my own original work except where acknowledgement is made by name or in the form of a reference. The work contained herein has not been submitted in part or whole for a degree at any other university.

Signed: 

Ariyo Sunday Ajibola

Date: July, 2013

As the candidate's supervisor, I have approved/disapproved the dissertation for submission

Signed: 

Dr. A. O. Adewumi

Date: July, 2013

UNIVERSITY OF KWAZULU-NATAL

COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE

DECLARATION – PLAGIARISM

I, **Ariyo Sunday Ajibola**.....declare that

1. The research reported in this thesis, except where otherwise indicated, is my original research.
2. This thesis has not been submitted for any degree or examination at any other University.
3. This thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
4. This thesis 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 and 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 thesis does not contain text, graphics or Tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the References section.

Signed:

Ariyo Sunday Ajibola

Table of Contents

List of Tables

List of Figures

List of Acronyms

ACO:	Ant Colony Optimisation
AIMMS:	Advanced Interactive Multidimensional Modeling System
BAP:	Berth Allocation Problem
BPP:	Bin Packing Problem
BSAP:	Berth Space Allocation Problem
CA:	Category Allocation
COP:	Combinatorial Optimisation Problem
Ds:	Discretionary students
FA:	Floor Allocation
Fo:	Foreign students
Fr:	Fresher (Fresh students)
Fy:	Final year students
GA:	Genetic Algorithms
GAP:	Generalised Assignment Problem
GDA	Generalized Deterministic Annealing
GS:	Global Search
GUI:	Graphical User Interface
HA:	Hall Allocation
HC:	Hill Climbing
HIL:	Higher institution of Learning
HSAP:	Hostel Space Allocation Problem
Ht:	Health students

IDE:	Integrated Development Environment
IPOPT:	Interior Point OPTimizer
KNITRO:	*N*onlinear *I*nterior point *T*rust *R*egion *O*ptimization (the "*K*" is silent)
KSP:	Knapsack Problem
LAHC:	Late Acceptance Hill Climbing
LP:	Linear Programming
LS:	Local Search
NLP:	Non-Linear Programming
NP-hard:	Non-deterministic Polynomial-time Hard in computational complexity theory
OSAP:	Office Space Allocation Problem
Ot:	Other students
PSO:	Particle Swarm Optimisation
SA:	Simulated Annealing
SAP:	Space Allocation Problem
Sc:	Scholar students
SNOPT:	Sparse Nonlinear OPTimizer
Sp:	Sport men and women
TA:	Threshold Accepting
TS:	Tabu Search
TTP:	Time Tabling Problem
XML:	Extensible Mark-up Language

Abstract

This research work focused on the performance of heuristics and metaheuristics for the recently defined Hostel Space Allocation Problem (HSAP), a new instance of the space allocation problem (SAP) in higher institutions of learning (HIL). SAP is a combinatorial optimisation problem that involves the distribution of spaces available amongst a set of deserving entities (rooms, bed spaces, and office spaces etc.), so that the available spaces are optimally utilized and complied with the given set of constraints.

HSAP deals with the allocation of bed space in available but limited halls of residence to competing groups of students such that given requirements and constraints are satisfied as much as possible. The problem was recently introduced in literature and a preliminary, baseline solution using Genetic Algorithm (GA) was provided to show the viability of heuristics in solving the problem rather than recourse to the usual manual processing. Since the administration of hostel space allocation varies across institutions, countries and continents, the available instance is defined as obtained from a top institution in Nigeria. This instance identified is the point of focus for this research study. The main aim of this thesis is to study the strength and performance of some Local Search (LS) heuristics in solving this problem. In the process however, some hybrid techniques that combine both population-based and LS heuristics in providing solutions are derived. This enables one to carry out a comprehensive comparative study aimed at determining which heuristics and/or combination performs best for the given problem.

HSAP is a multi-objective and multi-stage problem. Each stage of the allocation has different requirements and constraints. An attempt is made to provide a formulation of these problems as an optimisation problem and then provides various inter-related heuristics and meta-heuristics to solve it at different levels of the allocation process. Specifically, Hill Climbing (HC), Simulated Annealing (SA), Tabu Search (TS), Late Acceptance Hill Climbing (LAHC) and GA were applied to distribute the students at all the three levels of allocation. At each level, a comparison of the algorithms is presented. In addition, variants of the algorithms were performed from a multi-objective perspective with promising and better solutions compared to the results obtained from the manual method used by the administrators in the institutions. Comparisons and analyses of the results obtained from the above methods were done.

Obtaining datasets for HSAP is a very difficult task as most institutions either do not keep proper records of past allocations or are not willing to make such records available for research purposes. The only dataset available which is also used for simulation in this study is the one recently reported in literature. However, to test the robustness of the algorithms, two new data sets that follow the pattern of the known dataset obtained from literature are randomly generated. Results obtained with these datasets further demonstrate the viability of applying tested operations research techniques efficiently to solve this new instance of SAP.

Acknowledgements

My deep appreciation goes to the Lord Jesus Christ for the guidance, inspiration, perfect health, protection, provision, preservation, vision, grace, mercy and above all love He has for me. I will forever remain grateful to you my God.

I would like to express my heartfelt appreciation to my supervisor, Dr. A.O. Adewumi, for his support in terms of his assistance, encouragement, guidance, financial support and interest in my work and progress. I am deeply indebted to you, sir for all your effort and most of all the time you have devoted to see to the completion of this thesis.

To my love, Aderonke and to my children, Esther and David for your prayers, support, patience and for the love you have bestowed on me during the course of this research work, I say thank you for always being there.

I would like to thank my mother who suffered to ensure that all her children are educated. Your years of sacrifices have borne fruits today. I pray that God will prolong your life to reap the reward of your labour. I would like to appreciate all my brothers and sisters for their unwavering support and encouragement.

Special thanks to the Faculty of Agriculture, Science and Engineering, faculty bursary and School of Mathematics, Statistics and Computer Science for creating an enabling and conducive environment.

I would also like to specially thank all the members of Deeper Life Bible Church, KZN province and UKZN's optimisation group for all their prayers and support.

CHAPTER ONE

INTRODUCTION AND BACKGROUND

1.1 Introduction

Hostel Space Allocation Problem (HSAP) is a huge source of concern for university administrations especially in developing countries where hostels are provided to students for residential purposes (Alitheia 2012; Adewumi & Ali, 2010, Adewunmi 2010). This concern stems from many conflicting factors. Primarily, due to the fact that university funding does not seem to favour the construction of more hostel facilities for the ever increasing population of students. Therefore, there is the need to manage available spaces efficiently to serve the needs of students while achieving the overall goal of the institutions. However, HSAP deals with the process of allocating a limited number of bed spaces within hostels (resources) among many competitive customers (eligible students) under a given set of hard and soft constraints (Adewumi, 2010). Adewumi & Ali (2010) defined HSAP as a combinatorial optimisation problem (COP) and proposed the use of heuristics to handle this new instance of the Space Allocation Problem (SAP).

Meanwhile, the range of optimisation methods that have been used to handle the COP consist of two main groups which are the exact (traditional) and approximate (heuristics) methods (Landa-Silva, 2003). Exact methods seek to find the optimal solution to an optimisation problem but have expensive computational needs that increase with growing difficulty of the problem. In addition, exact methods may not find solution to some real world problem. However, for practical applications, heuristic methods seek to find good solutions (in most cases near-optimal) within short computational times while trading accuracy for computational efficiency (Adewumi,

2010; Landa-Silva, 2003). Metaheuristics are improved heuristics methods that have been successfully applied to many COPs and SAPs for the past few decades.

This thesis reports a further investigation and study into the effectiveness of applying heuristics techniques to solve the HSAP within the context of a Higher Institution of Learning (HIL). A previous study on HSAP (see Adewumi & Ali, 2010) had advocated the application of optimisation techniques to handle the allocation process. This research study on the HSAP seeks to take the previous study further, both in terms of modelling and heuristic solutions. We provide a model for the last stage of the allocation process while also investigating the performance of more heuristics, metaheuristics and their hybridizations in solving this instance of SAP. Extensive simulation studies were carried out on five combined heuristics techniques (and hybrids) with their comparative results reported.

Before giving a further overview of the problem, a brief general overview of an optimisation problem is presented.

1.2 General Optimisation Problem

Optimisation is the field of study that seeks to obtain the best possible results under given constraints and several alternatives. Optimisation problems abound in all fields of study and all areas of human endeavour including engineering, science, technology, aeronautics and even in planning warfare. The main objective is to seek to optimise (maximize or minimize) certain decision variables. Mathematically, therefore, optimisation seeks to find the best value for decision variable(s) that would maximize or minimize an objective function. Optimisation

techniques provide procedural steps that help in exploring a search space to seek a feasible solution that optimises given objective functions under stated constraints. It is a known fact that there is no single optimisation technique that can provide optimal solutions to various forms of optimisation problems hence the search for better techniques as well as the improvement/application of known techniques to solve new problems are the subjects of on-going research activities. Literature has proposed several optimisation methods or tools for solving diverse types of optimisation problems.

Definition 1.1

If we assume x^* to be a decision variable, then an optimisation problem can be stated as:

$$\max g(x) \tag{1.1}$$

subject to $x \in S$ (1.2)

$$g(x) \leq g(x^*) \text{ for all } x \in S \tag{1.3}$$

where x^* maximises the function g subject to the constraint $x \in S$, and that $g(x^*)$ is the maximum value of the function g subject to the constraint $x \in S$. Minimization is simply a negation of maximization, i.e [min f(x) = -max {-f(x)}].

Definition 1.2

The variable x* is defined as a local maximiser of the function g subject to the constraint x∈S if there is a number σ > 0 such that g (x) ≤ g (x*) for all x∈S for which the distance between x and x* is at most σ. A local minimiser is also defined similarly.

Optimisation problems can be classified based on several factors and characteristics of the problem to be solved. A few classifications of optimisation problems according to certain characteristic features are presented in the next sub-section.

In a broad sense, optimisation problems can be solved using exact (traditional) and/or heuristic techniques. Exact methods seek to obtain an optimal solution to a given problem but in most cases at the expense of computational time especially for many NP-Hard problems. They may therefore not be too appropriate for some complex and difficult problems (Adewumi, 2010). Heuristic (approximate) algorithms, on the other hand, seek to get near-optimal solutions to given optimisation problems thereby compromising accuracy for computational speed. However, attempts geared towards seeking a balanced trade-off between accuracy and significant reductions in computational time are currently being made.

1.3 Classification of optimisation problems

1.3.1 Classification based on constraints

Constraints are the limits that restrain the value of the objective function *g*. They characterize the bounds within which feasible solutions are obtained. Constraints can be of two types: hard constraints or soft constraints. Hard constraints define the feasibility of the solutions to be obtained and cannot be violated while soft constraints can add to the quality of the solutions but can be compromised with or without penalty. In a general sense, all hard constraints must be satisfied and as many soft constraints as possible need to be satisfied if one is to get any feasible

solutions. Hence, based on the types of constraints, optimisation problems can be categorised into unconstrained and constraint optimisation problems (Rao, 1996).

a) **Unconstrained Optimisation Problems**: If there are no constraints leading to the evaluation of *g*, the problem is considered an unconstrained optimisation problem. If *m* equals the number of constraints, then *m*= 0.
b) **Constrained Optimisation Problems**: If there are constraints leading to the evaluation of *g*, the problem is considered to be a constrained optimisation problem. If *m* equals the number of constraints, then $m \geq 1$. Most of the real-world optimisation problems are multi-constrained problems.

1.3.2. Number of objective functions

Based on the number of objective functions to be minimized, optimisation problems can be categorised into two, viz. single and multi-objective programming problems.

Single-Objective Programming Problem:

A single-objective programming problem can be described as the following:

Find *x* which minimizes $g_1(x)$ (1.4)

subject to

$$g_j(x) \leq 0,\ J= 1,2,...,m \quad (1.5)$$

where g_1 denotes the objective functions to be minimized.

Multi-objective Programming Problem:

A multi-objective programming problem can be described as the following:

Find *x* which minimizes $g_1(x), g_2(x), \ldots, g_k(x)$ (1.6)

subject to

$$g_j(x) \leq 0,\ J= 1,2,...,m \tag{1.7}$$

where g_1, g_2,..... g_k denote the objective functions to be minimized simultaneously.

1.3.3 Nature of the problem

Another significant classification of optimisation problems is done on the basis of the nature of expressions for the objective function and the constraints. In line with this classification, optimisation problems can be categorised into various forms: linear, nonlinear, geometric, and quadratic programming problems.

- **Linear programming**

  Linear Programming (LP) problem is such that both objective functions and constraints are linear functions of the design parameters. ALP problem is often described in the following standard form:

  Find $x = (x_1, x_2, ..., x_n)$ (1.8)

  which minimizes $g(x) = \sum_{i=1}^{n} c_i x_i$ (1.9)

  subject to constraints

  $$\sum_{i=1}^{n} a_{ij} x_i = b_j, j = 1, 2, \ldots\ldots, m \tag{1.10}$$

  $$x_i \geq 0, i = 1,2, \ldots\ldots, n. \tag{1.11}$$

  where c_i , a_{ij} and b_j are constants.

- **Nonlinear programming**

  When there are a number of variables determining the objective and constraint functions, the problem is termed a Non-Linear Programming (NLP) problem. This type of problem is quite common, other problems can be considered as particular cases of the NLP problem.

- **Geometric programming**

  A function v(x) is termed as a polynomial if *v* can be expressed as the sum of power terms each of the form

$$c_i x_1^{a_{i1}} x_2^{a_{i2}} \dots\dots\dots x_n^{a_{in}} \tag{1.12}$$

  where c_i and a_{ij} are constants with $c_i>0$ and $x_j>0$.

  So an N term polynomial is expressed as the following

$$v(x) = c_1 x_1^{a_{11}} x_2^{a_{12}} \dots . x_n^{a_{1n}} + \dots \dots + c_N x_1^{a_{N1}} x_2^{a_{N2}} \dots . x_n^{a_{Nn}} \tag{1.13}$$

  A geometric programming problem is one in which the objective function and constraints are expressed as polynomials of x .

- **Quadratic programming**

  A subset of NLP problems with a quadratic objective function and linear constraints is known as quadratic programming which can be depicted as follows:

$$g(x) = c + \sum_{i=1}^{n} q_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} Q_{ij} x_i x_j \tag{1.14}$$

  subject to

$$\sum_{i=1}^{n} a_{ij} x_i = b_j, \quad j = 1, 2, ..., m, \tag{1.15}$$

$$x_i \geq 0, \quad i = 1, 2, ..., n, \tag{1.16}$$

  where $c, q_i, Q_{ij}, a_{ij},$ and b_j are constants.

1.3.4Nature of the decision variables

Optimisation problems can be categorised as continuous or combinatorial optimisation problems based on the decision variables used.

- **Continuous Optimisation Problems:**

The model form of an (continuous) optimisation (Boyd & Vandenberghe, 2004) problem is:

$$\min f(x) \tag{1.17}$$

subject to

$$g_i(x) = 0, \quad i = 1, ..., m \tag{1.18}$$

$$h_i(x) = 0, \quad i = 1, ..., p \tag{1.19}$$

where

- $f(x) : \Re^n \to \Re$ is the goal function which is minimise over the variable x,
- $g_i(x) \leq 0$ are called inequality constraints, and
- $h_i(x) = 0$ are called equality constraints.

- **Combinatorial Optimisation Problem (COP):**

COP is in quadruple form (I, f, m, g), where:

- I is defined as the set of instances;
- When an instance is provided $x \in I$, $f(x)$ is the possible solution set;
- When an instance is provided x and the possible answer is y or x, $m(x, y)$ stand for the evaluation of y, that is more often than not a positive real.
- g is the objective function which is either min or max .

The objective can then be defined as being to discover some case x as an optimal solution, which is a feasible solution y with

$$m(x, y) = g\{m(x, y') \mid y' \in f(x)\} \tag{1.20}$$

For every COP case, there is a consequent decision problem that hinges on the fact that there is a feasible solution for some particular measure m_0.

1.4. Heuristics and Metaheuristic Algorithms

Heuristic optimisation algorithms (heuristics for short) search for good feasible solutions to optimisation problems in situations where the complexities involved or the paucity of time does not permit an optimal solution (Garey & Johnson, 1979). Unlike exact algorithms, there are two very strong issues that have to be considered in the evaluation of heuristics, they are: how quickly solutions can be obtained and how close they are to being optimal.

A metaheuristic is a black box process that guides a subordinate heuristic by merging cleverly different concepts for investigating and exploiting the search space in order to obtain efficient near-optimal solutions (Osman & Laporte, 1996). Examples include SA, TS and GA (Kirkpatrick et al., 1983; Holland, 1975), most of which are inspired by social behaviour or concepts in nature. Metaheuristic algorithms try to strike a balance between exploring and exploiting the local neighbourhood structures of the solution space (Syam & Al-Harkan, 2010).Exploitation involves locating more ‘promising’ local neighbourhood structures as these areas may enclose superior solutions while exploration seeks to find the global optimum solution point(Syam & Al-Harkan, 2010). This leads to the classification of optimisation search techniques into LS and Global Search (GS) techniques.

LS techniques work by applying local changes of some sort within a defined neighbourhood contained in the search space, until a solution deemed optimal is found or a time frame/limit has elapsed. These methods have proved highly effective in solving some optimisation problems and very recently as supportive cum improvement hybrid to GS techniques (Battiti, Brunato & Mascia, 2008). Examples of LS methods include Hill Climbing (HC), Guided Local Search and Iterative Local Search algorithms, among others. GS techniques, on the other hand, aim to find the global optimum solution within the best possible value(s) of decision variables within the GS space by way of effective combination of exploitation and exploration features of the underlying algorithm. However, there are many real-world problems in which locating the global optimal solution still remains practically impossible (Landa-Silva, 2003; Wikipedia, 2012). Furthermore, there is no single GS technique that guarantees locating the global optimal solution to all kinds of optimisation problems hence the need for simulation experiments to sometimes determine which technique is best for a given state of an optimisation problem. GS heuristic algorithms, therefore, only attempt to assess the global optimal solution from a set of local optimal solutions. Examples of GS techniques include GA, Particle Swarm Optimisation (PSO), Ant-Colony Optimisation (ACO), among others.

1.5. Objectives of the Study

It is practically difficult to handle Non-deterministic Polynomial-time hard (NP-Hard) real world COPs with the exact solution techniques especially where the search space is considerably large, hence, most researchers settle for near-optimal solutions that provide realistic solutions to these problems. SAPs recently became an interesting research area in metaheuristic research with

various interesting case instances being explained in literature (for example, see Landa-Silva, 2003; Burke, Cowling & Silva, 2001; Adewumi & Ali, 2010).

SAP addresses the challenge of allocating limited available space among a set of demanding entities requiring space utilization. These classes of problems are multi-constrained and multi-objective problems in nature (Adewumi, 2010). Finding feasible solutions to them require the maximization of space utilization in such a way that satisfies all hard constraints while satisfying as many soft constraints as possible (Adewumi, 2010). The aim is to provide as much satisfaction as possible to all demanding entities that require space utilization. Instances of the SAP that have been introduced in literature especially as they relate to HILs include the Office Space Allocation Problem (OSAP), timetabling problems and most recently, parking spaces allocation problems and the HSAP.

The HSAP, a recent instance of SAP in literature, is concerned with the efficient allocation of limited amounts of bed spaces to eligible students within the halls of residence at a given HIL. HSAP is an NP-hard COP and like other SAPs is not only multi-constrained and multi-objective but also multi-staged hence near-optimal solutions are determined using heuristic and metaheuristic algorithms (Adewumi & Ali, 2010).

The objective of this research study is to further investigate the deployment of heuristics in solving the HSAP. Our emphasis is on investigating and comparing the effectiveness of LS heuristic algorithms in providing solutions to the HSAP. The research study investigates five techniques namely, HC, Simulated Annealing (SA), Tabu Search (TS), Late Acceptance Hill

Climbing (LAHC) and GA. As an extension, some hybrids of these algorithms are investigated with other search metaheuristic algorithms. Results obtained from these different instances are compared.

1.6. Thesis Outline

The chapters of this thesis are organized as follows: Chapter 2 introduces SAP at HILs and explains some of the problems experienced in utilizing space efficiently at these institutions. It also reviews some variants of the SAP. Chapter 3 introduces the HSAP. It describes the HSAP and discusses previous research that has been done in this area. It also presents the mathematical modelling of the problem. Chapter 4 presents the methodology employed in providing solutions to the problem. Chapter 5 presents and discusses the experimental results obtained. Chapter 6 contains the conclusion and discussions of future extensions to the problem. Appendices A, B, C and D provide details about the three data sets used as well all the associated constraints involved in the allocation process of the HSAP.

1.7. Contributions

The following are the contributions of the thesis:

1. **The thesis explores the performance of the various heuristics which are most common in literature and also the performance of their hybrids which is considered as the strong point of this research work.**
2. The thesis provides new datasets on which further research o n the HSAP can be based.
3. Several important issues in the HSAP were identified, and practical models for this type of HSAP are proposed including the floor stage.

CHAPTER 2

SPACE ALLOCATION PROBLEMS

2.1. Introduction

SAPs are one of the most difficult NP-hard COPs (Adewumi & Ali, 2010; Burke and Varley, 1998) to solve. It is a very important space management issue that is concerned with the distribution of limited available space amongst demanding sets of entities requiring space utilization. It is a well-known fact that space available to accommodate entities (for instance, bed space for students, shelf space for items) is often limited especially as an organization grows. Mismanagement of this limited space can thus adversely affect the overall operations of an organization. Inefficient use of the limited spaces can in turn affect the overall costs involved in the organisation's operation, amongst others. Since the cost and feasibility of space expansion is often and almost impracticable in many real life situations, it is pertinent therefore to consider best practices in the management of available space. Although, a strict management issue, space allocation is essentially an optimisation problem and can thus benefit from mathematical modelling and optimisation research, hence the significance of this research study. It is often easy to conceive SAP in terms of well-known COPs as benchmark problems like bin-packing problems, knapsack problems, etc. are essential space utilization or allocation problems.

In this section, we give a general description of space allocation with specific focus on some instances of the SAP such as Berth Space Allocation Problem (BSAP), Office Space Allocation Problem (OSAP) (Pereira, et al., 2010), Timetabling Problem (TTP) (Adewumi, Sawyerr & Ali, 2009; Burke & Bykov, 2008 & 2010) and HSAP. This section describes some of the problems

and complexities involved in addressing space utilization, and give an overview of some instances of SAP as well as more details on the HSAP.

2.2. Space Allocation in Higher Institution of Learning

Space utilization presents a common challenge to HILs. This is due to the challenge of distributing limited numbers of available space among demanding entities that require space utilization. The demanding entities may include staff, lecture venues, students demanding on-campus accommodation, laboratories or practical etc. (Adewumi & Ali, 2010; Landa-Silva, 2003).In addition, the objective of space allocation is to provide optimum satisfaction to all demanding entities while satisfying all hard constraints and as many soft constraints and requirements as possible (Adewumi & Ali, 2010).

Mismanagement of available space in HILs may negatively affect the overall running and operating costs of the institution hence the need for effective and efficient utilization and management of space. However, finding optimal solutions in the way space is utilized presents a challenge as SAPs are computationally "hard" in nature. Furthermore, the problem is complicated due to the dynamic nature of space management in real life instances as entities are added and removed continuously (Landa-Silva, 2003). Also, in determining the best solution to SAPs, the convenience of the underlying entities may be an important factor. For example, faculty and departments should be allocated spaces as close as possible to lecture venues while physically challenged students need to be allocated hostel space that is closest to health facilities, amongst others. These issues make space allocation a very important managerial responsibility and hence automated solutions that incorporate good approximation algorithms are essential.

This will provide for efficient and effective, accurate and fair distribution of spaces without personal biases. However, many institutions, particularly in developing countries, still rely on using manual processes in dealing with space allocation at HILs.

The manual approach, though in some cases may rely on some form of computer processing which are prone to inefficiency, errors and biases. From experience, the timely production of distribution lists also poses a great challenge since there is no guarantee that solutions obtained via this approach will either be good or near the expected optimal solution. While the manual approach may be relatively easy and quick for small sized organizations, dealing with cases involving larger population sizes such as the allocation of hostel space in higher institutions will pose a challenge. This is due to the larger sizes of the input data sets and the complexities of the constraints and objectives associated with obtaining solutions.

Mathematically, most SAPs have been modelled using well-known benchmark COPs such as BPP, Knapsack Problem (KSP), assignment, or resource allocation problems. Similarly, this study employs a form of multiple knapsack models to model the different stages of the HSAP. A brief discussion of some of these benchmark models are given below while details of the HSAP is presented in the Chapter 3.

2.3. Benchmark Model commonly used for SAP

2.3.1 Knapsack Problem

KSP is a very common NP-Hard benchmark COP that has been used in modelling many real-world optimisation problems. It involves the arrangement or assignment of items (or the subsets thereof) into knapsack(s) so as to maximize the total accumulated profits of the items while the capacity constraint of the knapsack(s) is/are being observed. Each item to be arranged has associated profit and weight values as illustrated in the simple example in Figure 2.1.

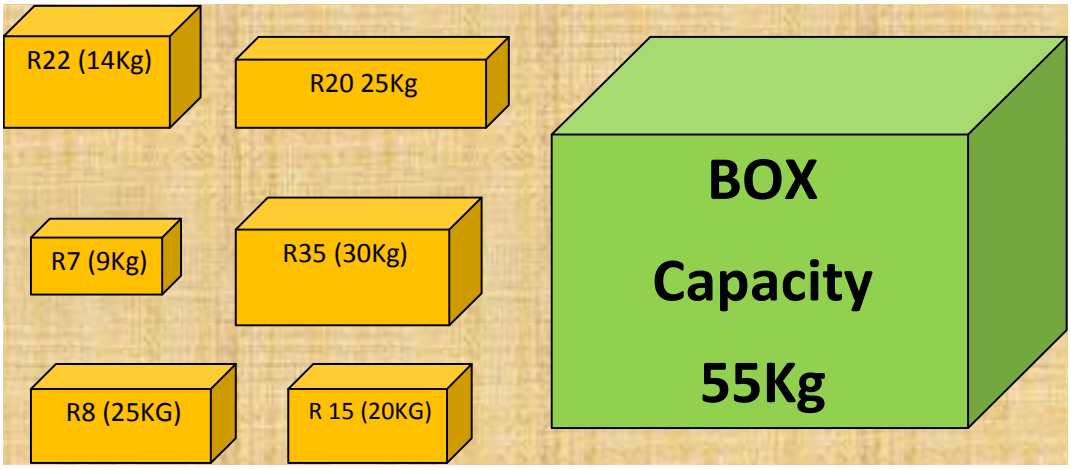

Figure 2.1: An illustration of a one-dimensional knapsack problem (Source: Wikipedia)

Various forms of knapsack models (and their variants) have been applied in literature to model optimisation problems. These include the binary, fractional, bounded, multiple, and quadratic knapsack models (Landa-Silva, 2003; Martello & Toth, 1990a).The differences lie in the way the items are distributed and the number of knapsacks involved (Nyonyi, 2010). In a binary model, an item is either selected or not selected, while in the fractional model, a fraction of items can be selected. The bounded knapsack model allows for an upper bound on the number of times an item can be selected while the multiple knapsacks has more than one knapsack where the items can be placed. The latter can be binary, fractional or any other combination.

The aim of the multiple knapsack modelling is to fill multiple knapsacks with subsets of items in such a way that the total accumulated profits of the subsets are maximized without having the total accumulated weights of the subsets exceed the capacities of the knapsacks. Mathematically, a binary otherwise called the 0-1 multiple knapsack model can be described as follows:

Let

m = number of knapsacks

n = number of items

c(i) = capacity of knapsack i

p(j) = profit associated with item j

w(j) = weight associated with item j

x(i,j) = 1 if j is selected for knapsack i, 0 otherwise

The objective function is:

Maximize

$$f(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} p(j)x(i, j) \tag{2.1}$$

subject to:

$$\sum_{j=1}^{m} w(j)x(i, j) \leq c(i) \quad i = 1, 2....., m \tag{2.2}$$

$$\sum_{i=1}^{n} x(i, j) \leq 1 \quad \text{j} = 1, 2....., n; \qquad \{0,1\} \tag{2.3}$$

In this research study, a form of the 0-1 multiple knapsack model is used to model the different stages of the HSAP.

2.3.2 Bin packing problem

The BPP is another well-known NP-Hard COP with various forms appearing in literature as well (Martello & Toth, 1990a). We defined one dimensional bin packing problem as follows: Given a set of entities or items $I = \{1, ..., n\}$ each having a corresponding size or weight w_i and a set of bins with identical capacities c. The goal is to pack all the items into a few bins while observing the size constraint of the bins. Many researchers have studied different dimensions of the BPP and applied them to modelling and to solve real-world problems using both exact and heuristics techniques (for example see Scholl, Klein & Jurgens, 1997; Martello & Toth, 1990a).

2.3.4 Generalised Assignment Problem

The Generalised Assignment Problem (GAP) is an NP-Hard problem that is similar to the multiple KSP except that the profit and weight of each item varies with respect to the containers assigned to them (Burke et al., 2000). It can be mathematically formulated as follows:

m = the number of containers;

n = the number of items;

p_{ij} = profit of item i if allocated to container j;

w_{ij} =weight of item i if allocated to container j;

c_j = capacity of container j

$$x_{ij} = \begin{cases} 1 \text{ if item } i \text{ is assigned to knapsack } j, \\ 0 \text{ otherwise.} \end{cases}$$

$$\max \quad \sum_{j=1}^{m} \sum_{i=1}^{n} p_{ij} x_{ij} \tag{2.4}$$

$$\text{subject to } \sum_{i=1}^{n} w_{ij} \leq c_{j,} \quad j = 1, \ldots, n \tag{2.5}$$

$$\sum_{j=1}^{m} x_{ij} \leq 1, \qquad i = 1, \ldots, n \tag{2.6}$$

$$x_{ij} = 0 \text{ or } 1, \quad i = 1, \ldots, n, \; j = 1, \ldots, m \tag{2.7}$$

A practical application of the model is assigning n tasks to m processors, (or n jobs to m machines) given the profit p_{ij} and the level of resource required w_{ij} for the assignment of task i to processor j and total resource c_j available for each processor j (Martello & Toth, 1975; Martello & Toth, 1990a).

2.4. Related Works

On-campus residence for students of HILs is a very pertinent issue as their availability and efficient management have been shown to influence the performance of students (M&G 2009; Oghifo 2012). Many irregularities and strikes in HILs across Africa have been linked to the problem of unavailability and/or mismanagement of residential accommodation for students. The above are important reasons why it is necessary to use automated systems to assist in finding effective solutions for allocating students accommodation. Automated systems are fairer, more accurate and faster, compared to manual processes. However, many HILs still employ manual processes. This is primarily prevalent in developing countries (Adewumi & Ali, 2010).

Meanwhile, this problem was recently described as a COP and heuristics solutions have been proposed in solving it (Adewumi, 2010). Earlier, metaheuristics have been successfully applied to similar SAPs. The use of heuristics (and its variants) was necessitated by the complex nature of the problem for which exact algorithms have proved insufficient especially as the problem's search space increases. Researchers have therefore advocated the use of heuristics and any efficient hybrid thereof for SAPs. Previous studies have employed both population-based and LS techniques as well as their hybrids (e.g. Math-heuristics, hyper-heuristics) to solve SAP. The initial study on HSAP was essentially based on GA. This current study therefore looks further into the performance of LS and hybrids of the HSAP.

Generally, space planning is a major issue in HILs just as management has to cope with the demand for office space, lecture venues, examination venues, residence space etc. There are many factors that influence how available spaces are allocated with each HIL differing from another in terms of its space planning policy and management. Previous research concentrated on these areas of space allocation in HIL: office space (Burke, Cowling, & Silva, 2001; Fomeni, 2010; Landa-Silva., 2003;Pereira, et al., 2010;Silva, Ferreira, & Costa, 2008) and timetabling (Adewumi,Sawyerr& Ali, 2009; Burke & Bykov , 2008 & 2010).Current research efforts are a furtherance of the recent focus on hostel space allocation.

Past studies on SAP sought to formulate mathematical models for the identified real-world instances while considering various constraints and requirements. The issue of hostel space has great influence not only on the academic performance of students (Alitheia, 2012; Pat-Mbano,

Alaka1 & Okeoma, 2012; Yusuff, 2011) but also on their safety, convenience and undivided concentrations on their primary duty. In Nigeria where the current case study is based, some real estate agents have started considering the economic potential of developing private student housing near university campuses both to serve the needs of students while carrying out their own professional services (Alitheia, 2012). Though this proposition was welcomed with some controversies, many major HILs have witnessed private residence provision by realtors, private landlords and other stakeholders that have landed properties close to campuses (Pat-Mbano, Alaka1, & Okeoma, 2012). Thishas compelledadminstrators of HILs in the country to rethink the management strategies for the distribution of available hostel space, this is the major motivation/justificationfor this study.Some other instances of SAP are briefly described below.

2.4.1. Berth Space Allocation Problem

Berth Space Allocation Problem (BSAP) is a commonly studied SAP in literature. The BSAP seeks to assign a set of vessels to a given berth layout within a given time horizon. Appropriate allocation and positioning of ships carrying containers has been a major source of concern for a long period of time. Fluctuations in the demand for ships carrying containers have created considerable apprehension leading to serious optimisation problems at the marine terminals. In addition to dealing with space allocation, the BSAP also considers time (temporal dimension) as a major constraint. Depending on the case and instance at hand, there can be several objective functions to be optimised, for example, to minimize the service time to vessels, minimize the time of stay at the port, or to minimize the number of rejected vessels. Each instance and real life case of the BSAP has varying constraints and requirements such as the spatial and temporal constraints. Therefore, there is no unique mathematical model that can fully describe the BSAP,

it depends on the case, objective and constraint at hand. Several models have been reported in literature involving BSAP with various temporal (such as vessel arrival process, start of service and handling times) and spatial (such as berth layout and restrictions) attributes. Regardless of the formulation of the problem, BSAP is an NP-hard or NP-complete problem that requires the use of heuristics and meta-heuristics to obtain solutions within reasonable computational time.

Most recently, Umang et al. (2013) studied the application of exact methods based on mixed integer programming and heuristics approach to solve the BSAP in bulk ports with the objective of minimizing service times of vessels for a given yard layout. The study, which was based on real life data, found that near-optimal solutions can be obtained for even larger instances with the heuristics. In addition, various exact and heuristics methods have been successfully applied to solve varying models of the BAP. Initially, queuing models were developed to solve the BAP (Edmond & Maggs, 1978) which is formulated as a COP. Various models of the BAP are presented in Buhrkal et al, 2010. Solution methods in literature include the exact or mathematical programming approaches (Umang et al, 2013), heuristics based on Lagrangian relaxation (Akio et al., 2001), use of clustering search (Oliveira et al., 2011)and hybrid approach of TS and mathematical programming (Giallombardo et. al., 2010), among several others.

2.4.2. Office Space Allocation Problem (OSAP)

The OSAP applies not only in HIL but also in many large organizations where the allocation of buildings and office space to departments, units, and employees pose a challenge especially given the increasing number of constituents demanding limited office space. In other words, OSAP seeks to assign employees workspace in an office building in an optimal way which

satisfies the given objective criteria and requirements. This can occur in two forms: a complete reassignment of all employees in the organization to a new workspace, this is likely due to reorganization or relocation; and secondly, a re-assignment or new assignment of workspace due to a change in the employees' composition such as new hiring and personnel turnover. The former might aim to maximize the use of available space while the latter might emphasise minimizing the disruption of the current workforce. Cases of multiple objectives are possible.

OSAP has been considered and modelled as a variant of the Bin Packing Problem (BPP), the Knapsack Problem (KSP) or the Generalized Assignment Problems (GAP) (Burke & Varley, 1998), which are well-known NP-complete problems in nature. Exact methods have been employed to solve instances of OSAP (for example, see Ulker & Landa-Silva, 2010a&b). Ulker & Landa-Silva (2010a) developed a 0/1 integer programming technique to solve the OSAP with the primary aim of optimizing space utilization while satisfying a set of given constraints. The model was solved using CPLEX with significant results obtained for some combinations of hard and soft constraints. In another related work by the same authors (Ulker & Landa-Silva, 2010b), a 1/0 integer programming formulation model was develop for OSAP using University of Nottingham's data set. The model was implemented using the Gurobi solver which gave a better result when compared with known result from the same data set.

Furthermore, various heuristic and meta-heuristic techniques have also been proposed including the GA, SA, TS, Particle Swarm Optimisation (PSO) and other hybrid approaches (see Landa-Silva & Burke, 2007; Landa-Silva et al., 2010; Ozg¨ur & Landa-Silva, 2012; Ulker & Landa-

Silva, 2012, Zahiri, 2009). For instance, Pereira et al. (2010) studied the performance of a greedy search and TS for generating high quality solutions to the OSAP with the objectives of maximizing synergies within the organization, minimizing over-usage of limited office space while also maximizing the number of closed spaces. The TS gave better performance than the greedy LS algorithm. A study by Lopes & Girimonte (2010) showed that extensions of a combination of LS operators can improve the performance of LS algorithms for this type of problem.

2.4.3. Timetabling Allocation Problem (TTP)

TTP is a major academic problem that has posed challenges to HILs worldwide. It exists in various forms of which are the lecture or course TTP and examination TTP, each with varying complexity of constraints. Course TTP involves scheduling a number of students taking given course(s), lecturers and lecture rooms into a fixed set of timeslots for days of the week in an optimal schedule. TTP generally has diverse set of constraints, resources and requirements depending at times on the different real-life scenario considered (Adewumi, et al., 2009). Some hard constraints common to TTP include having a schedule where no lecturer, class or classroom is used more than once in any given period. Both generic and real-life forms of TTP present various forms of hard and soft constraints (see Adewumi et. al, 2009; Murray, Uller & Rudov, 2010).

Similar to other SAPs, both exact and heuristics approaches have been successfully applied to solve the TTP.Landa-Silva & Obit (2011) designed constructive hybrid heuristics for the course

TTP. Four different hybrid heuristics that combine LS and the graph colouring method were tested with promising results shown by the constructive techniques. Schimmelpfeng & Helber (2006) modelled a case of examination TTP in Germany as a mixed integer assignment problem and found an exact solution using CPLEX solver. It was evident from this real world problem that an exact method can give satisfactory results if the problem size is not big White& Zhang (1998) employed a hybrid of TS and constraint logic to solve the course TTP for small data sets of a university timetable. Dammaket al. (2006) formulated TTP as a zero-one integer linear programming problem and applied a three-stage heuristic to solve it.

Burke & Bykov (2010) showed the effectivness of LS techniques in solving the TTP. Specifically, they designed LAHC to solve an instance of TTP with promising solutions. Although, HC has mostly been regarded as weak in handling large instances of COP, the study showed that an improvement on HC can be very promising. It therefore suggested the application of similar improved algorithm (LAHC, to be discussed later) to any problem where HC has been previously but unsuccessfully applied. This is why in this research study, the LAHC is used as one of the LS techniques in our HSAP. In addition, the authors further suggest the incorporation of improvement ideas into any search method where candidate and current costs can be compared.

2.5. Summary

High demand for on campus accommodation in the institutions of higher learning has initiated managers of these institutions to adopt systems to automate and optimise their decision making processes. The HSAP is one of the key factors for efficiency of any higher institution. Current

computer applications, however, do not offer hostel space allocation optimisation and humans usually do this intuitively.

This chapter gave an overview of the SAP and its importance. Also examples of specific cases where this problem has been tackled were given and these include: berth space allocation, office space allocation and TTP.

The chapter also describes some of the problems and complexities involved in SAP. However, these cases are discussed in more detail when addressing space utilization. Well-known mathematical models that have been used in the study of SAP are introduced, these include: the knapsack problem, bin packing problem and the generalised assignment problem.

This chapter raises several important issues about hostel space allocation. Due to the different constraints encountered in allocation, the HSAP can be a very difficult problem to solve.

However, due to the NP-Hard nature of the HSAP it is impractical to work out a polynomial time bounded solution procedure that can solve every problem instance to optimality. The first method proposed is dynamic programming used to optimise the space allocation model of which the hostel space problem is an example. However, this technique may require very high computational times for large problems. Heuristic and metaheuristic techniques were used as alternatives. This research work focuses on the heuristic and metaheuristic approaches in solving the HSAP.

CHAPTER 3

HOSTEL SPACE ALLOCATION PROBLEM

3.1. Introduction

As stated in the preceding chapter, the HSAP is an instance of the NP-Hard SAP (Adewumi & Ali, 2010). Like other SAPs, it is a multi-staged, multi-constrained, and multi-objective COP that involves finding feasible and acceptable solutions to the distribution of available but limited bed spaces in on-campus residences to eligible students in a way that satisfies given objectives. This chapter describes the HSAP and previous research that has been carried out in this area. Detailed background information associated with the problem is presented along with the different stages of the problem and the way in which solutions are found along with the mathematical formulations.

3.1 Hostel Space Allocation Problem

The adequate provision of student residence has its impacts on academic success as on-campus residence gives students peace of mind allowing them have consistent focus on their studies. The issue of residence provision for students has become a source of concern for administrators of HILs due to the increased pressure of students being admitted to such institutions without a corresponding increase in the provision of facilities. The demand for on-campus accommodation has increased significantly in recent years. This makes the management of student residences an important responsibility. With limited amounts of bed spaces available for accommodation, students need to be allocated in ways that are fair and as evenly distributed as possible. This

opens up a new phase of research for researchers especially in metaheuristic as the question of how best to manage and distribute available hostel space for students has not been adequately studied. The issue has been handled as a rather ad hoc process with varied degrees of success in different universities especially in developing countries (Adewumi & Ali, 2009). The pioneering research study reported in the HSAP (Adewumi & Ali, 2010) is based on experimental studies which were conducted using real-world data from one of the largest tertiary institutions in Nigeria. This data is representative of what obtains in other HILs in Nigeria but the situation might be slightly different in other countries including South Africa. However, efforts to gather more data from institutions in South Africa have yielded little or no results as most HILs do not keep proper records of such data. Some institutions have indicated that they are yet to initiate this proper recording keeping process. Furthermore, due to the novelty and nature of the problem at hand, there is as yet no method to measure the quality of hostel allocation and students distribution except against specified goals. The lack of data archives on previous allocations also makes it difficult to benchmark results with past manual results. Consequently, the success of space allocation was measured in terms of each of the variables involved, essentially the number of beds and the number of students that could be accommodated within the necessary guidelines of the university policy. The above limitation has constrained this study, being a further foundational research work in this area, to still be based on available data from the previous study as indicated earlier. The goal therefore is to further study the feasibility of other heuristics especially LS techniques in providing solutions to this problem. Moreover, the current study attempts to provide further mathematical modelling of the problem which was not very obviously presented in the previous study, hence the approach in this study is slightly different. It is an attempt to further show the viability and efficiency of heuristics in

tackling the HSAP. Since most HILs would require long-range plans to construct additional hostel space by building more hostels, it is imperative that the distribution of the existing space optimised to achieve given goals. Further studies on the HSAP based on the pioneer work done by Fomeni (2010) and Nyonyi (2010) have buttressed the efficiency of heuristics to the HSAP. Current research presents a mathematical model of the HSAP and re-applies GA techniques to solve it based on this model and a new chromosome representation. Other heuristics and their hybrids are also tested. A comparative study of results among the heuristics based on simulation experiments was conducted and reported.

3.1.1. Problem Description

In the case study on which this research is based (Adewumi & Ali, 2010; Adewumi, 2010), allocations of male and female students into hostels are done in a mutually exclusive manner as undergraduate hostels are delineated based on gender. From the dataset available, there are twelve on-campus residences with six designated for male and female respectively. Usually, residences are built as multi-story structures (with the exception of one hostel) each with varying numbers of floors that are further divided into blocks (otherwise call wings). Rooms are located on each wing per floor with each having one or more beds depending on the number of students it is designed to accommodate. Usually, due to the shortage of space, most rooms are designed to take more than one student and students on each wing have access to common facilities such as toilets and baths. The university, through the office of students' affairs, sets the criteria that make a student eligible for a bed space and each eligible student is entitled to only one bed space. The eligibility criteria may vary and is manually checked by staff.

After initial application and expression of interest in residence accommodation, the students' affairs officers classify all eligible students into categories for the purpose of allocation. Thus, the allocation of bed space goes through three distinct phases, each with different requirements and objectives. These are the category allocation, hostel allocation and floor allocation (discussed later in more detail). Once the distribution is done per floor, the porters in charge of each residence take charge of settling students into rooms and bed spaces. These stages are not essentially necessary as far as mathematical modelling and allocation distribution are concerned. With regards to this study, the categories are (Adewumi & Ali, 2010):

1. Final Year Students (Fy): Those in the last year of study
2. Scholars (Sc): Students with cumulative grade point averages that are in the first class range.
3. Foreign Students (Fo): whose nationality and residence is not Nigeria.
4. Health Students (Ht): Physically challenged students.
5. Fresher (Fr): First year and direct-entry students.
6. Sports students (Sp): Male and Female students who participate in sporting activities at the university.
7. Discretionary (Ds): Students considered based on special requests
8. Others (Ot)**:** All other students requiring accommodation (in various years of study)

Each category of students has peculiar characteristics and requirements which can be factored in as constraints into the allocation process. For example, disabled (health) students cannot be given allocation on the top floor in any residence since none of the hostels is built with escalators for ease of movement for them. Moreover, since the space available is limited, some of the

categories are prioritized based on pre-set administrative and/or other considerations (Adewumi & Ali, 2010; Nyonyi, 2010). This serves as a major hard constraint during category allocation. Other administative considerations that serve as either hard or soft constraints include: 1) Fy students must be allocated to a floor that will afford them less distractions (soft); 2) Ht students must be accommodated in hostels close to the medical centres and on the lowest floor for easy access (hard); 3) Sp students must be accommodated close to sports facilities due to practice (hard); 4) all Fy, Fo and Ht students should be accommodated (hard). Further details on the problem can be found in Adewumi (2010), Adewumi & Ali (2010) and Nyonyi (2010).

The process involved in the allocation problem is illustrated in Figure 3.1. As stated earlier, the allocation process is done in three stages namely: the category allocation (CA), Hall[1]Allocation (HA) and the Floor Allocation (FA) stages (Adewumi & Ali, 2010). The number of students to be accommodated are selected from each category at the CA stage while the HA stage seeks to distribute this students into various hostels. The FA stage distributes the students allocated to various hostels into various floors. The stages are interdependent as the output from the CA stage serves as input for the HA stage whose output also serves the FA stage. Each stage has varying constraints and requirements. Adewumi & Ali (2010) further sub-divided the eight categories of students into two for the purpose of modelling and solutions namely: fixed and flexible categories. The fixed categories are groups that must be accommodated at the CA stage, that is, all students in this group must be given bed space. At the HA stage, the fixed category represents those given preference in terms of distribution into various halls. The fixed category was determined based on given administrative criteria (mainly based on students' peculiarity) for

[1] It should be noted that the term hall is used as a synonym for hostel and these two terms would be used interchangeably throughout this research.

allocation. For example, at the CA stage, Ht, Fo and Sp must be accommodated hence they are considered fixed while at the HA stage, it is not necessary to give the Fo category special preference as they can be accommodated into any hostel. However, the Fy category are given preference at this stage due to the requirement to accommodate them where there will be less distraction. Moreover, the university has preferred specified hostels for final year students as at the time of this study (See appendix A for details).The flexible categories, on the other hand, are groups that are less restrained and are thus of lesser priority when it comes to hostel allocation. For instance, since there are always too many students than available space, only as many as possible of the flexible categories will be accommodated at the CA stage. Also at the HA stage, these categories can be distributed into any hall without restraint. The constraints for the HSAP can therefore be summarized as follows (Adewumi & Ali, 2010):

Hard Constraints:

1. The number of students accommodated must not exceed the total capacity of space available.
2. The number of students allocated to a specific hall must not exceed the capacity of that hall.
3. A student must be allocated only once (to one bed space).
4. At the CA stage, all fixed categories students must be accommodated.
5. At the HA stage, fixed categories must be allocated only to stated halls (see Appendix A).
6. Flexible categories must be allocated at the CA stage based on given priority.
7. At HA stage, Ht students should be allocated at the lowest possible floor.

Soft Constraints:

1. At the CA stage, as many Fy students as possible should to be accommodated.
2. Similarly, as many Fy students as possible should be accommodated.
3. At HA stage, Fy students should be accommodated at the highest possible floor.

The main goal is to achieve fairness in the distribution of available space.

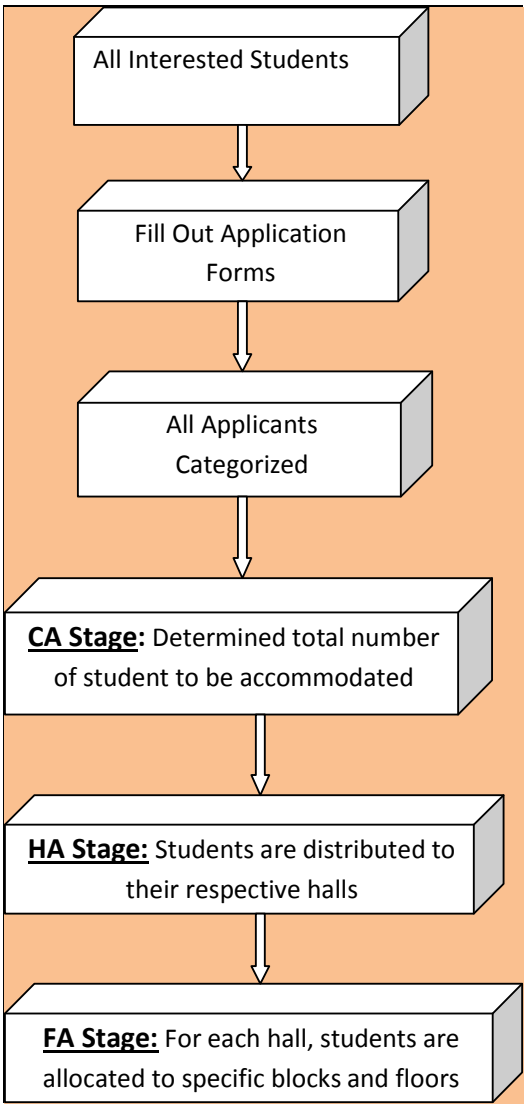

Figure 3.1: General HSAP Allocation Process

3.1.2 Data Sets (**Case Study)**

Secondary dataset on the number of hostels, sample number of applicants, categories of students and others were used as presented in Adewumi & Ali (2010) and shown in Appendix A. There

are twelve undergraduate hostels which are geographically spread across the main campus of the institution under study. The study did not deal with the postgraduate hostels which are separate from the undergraduate hostels and are administered based on different rules. The number of floors, wings and bed spaces (capacity) vary across the hostels as shown in Appendices A and B.

In order to test the efficiency of the models and algorithms used in this research, other random datasets that follow the distribution patterns of the available real dataset were generated. Two sets of data were generated and results were produced for each of them. The data sets have the same characteristics as our case study except that they were scaled to 1.5 of the original data set from the literature. In each of the data sets, we have eight categories of students and six halls of residence each for male and female students. In addition, each of the halls has varying capacities, blocks and floors. The generated data sets however follow the same requirements and constraints as specified for the original data set. See appendices A, C and D for detail.

A feasible solution to the HSAP is such as does not break any hard constraint and that satisfies as many soft constraints and objectives as possible. This needs to be done while allocating students in ways that are fair and that are as evenly spread throughout the halls of residence as possible. The models for the multi-staged allocation process are described in detail in the next section.

3.2. Modelling the Multi-stage HSAP

As earlier stated, the HSAP as defined in literature currently involves three stages of allocation namely the CA, HA and FA stages. The CA stage determines the exact number of students from each category that will be allocated accommodation. Once established, the HA stage determines the distribution of students from each category into specific halls while allocation to floors are done at the FA stage (see Figure 3.1).

3.2.1 Category Allocation Stage

As application for residence usually outnumbers the available space, this stage determines the students to be considered for accommodation from the eligible application pool in such a way that the total capacity of available space is not exceeded.Figure3.2 gives an overview of the allocation description at this stage which we modelled as a bounded KSP with restrictions. The modelling is described as follows:

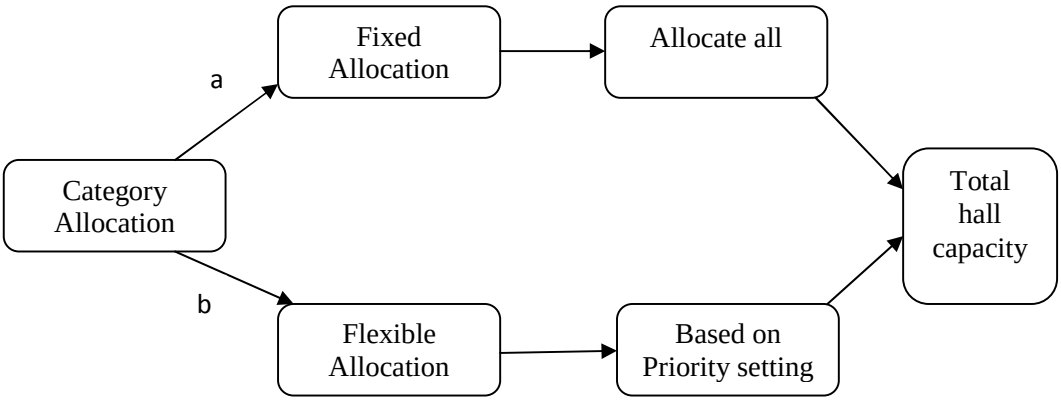

Figure 3.2: Description of Category Allocation Stage. Source: (Adewumi & Ali, 2010)

Applicants are classified into fixed and flexible categories of students as explained earlier with both groups assigned weights to represent the priority of allocation. We assume x_i to represent the number of students of i-th category to be allocated, c_i as the number of applicants in category i, k as the total count (number) of fixed categories, m as the total number of categories, T as the total number of allocations and Tv, as the total number of flexible allocations. A weight $w_i = 1$ is assigned to the fixed categories while weights $0 < w_i \leq 1$ is assigned to the flexible categories. The objective is to maximize the utilization of available space so as not to exceed the total available space. This is considered as

$$\max \sum_{i=1}^{m} w_i x_i \tag{3.1}$$

where w_i is the weight representing the allocation to students of category i among the set of applicants. Nyonyi (2010) assumed that $w_i = \frac{c_i}{\sum c_i}$ i.e. allocation is proportional to the number of applicants per category. However, in this study, we allow w_i to be user-defined as in another pioneer research by Adewumi & Ali (2010). The constraint is defined such that the full capacity of available space is used.

Subject to:

$$\sum_{i=1}^{m} x_i = T \tag{3.2}$$

This is an equality constraint rather than an inequality constraint which is only useful when the total number of applicants is smaller than the total capacity, in which case the solution of the problem becomes trivial. Besides, with the equality constraint, there is the guarantee that all spaces would be fully occupied. To take care of the fixed categories, we set

$$x_i = c_i,\ i = 1, ..., k \tag{3.3}$$

This constraint ensures that all students in fixed categories are allocated. However, for the flexible category, we have

$$0 \leq x_i \leq c_i,\ i = k+1, \ldots, m \tag{3.4}$$

This constraint ensures that allocated students in the category i are smaller than the number of applicants in category i. If the fixed categories are allocated, one is left with the fair distribution of the flexible categories, thus the objective becomes:

$$\max \sum_{i=k+1}^{m} w_i x_i \tag{3.5}$$

Subject to

$$\sum_{i=k+1}^{m} x_i = T_V \tag{3.6}$$

Feasible solutions determined at this stage will represent the number of students that will be given accommodation in the fixed and flexible categories. These students will then need to be distributed throughout the halls of residence at the HA stage.

3.2.2 Hall Allocation Stage

This stage follows and takes input from the CA stage. At this point, the distribution of students into halls based on fair and evenly distributed means are considered. Constraints for this stage have been discussed earlier and presented in Appendix A (Adewumi & Ali, 2010; Nyonyi, 2010). The fixed categories defined for this stage must be allocated to specified halls while the flexible categories are allocated to the remaining bed spaces in such a way that there is even distribution in all the halls as illustrated in Figure 3.3.

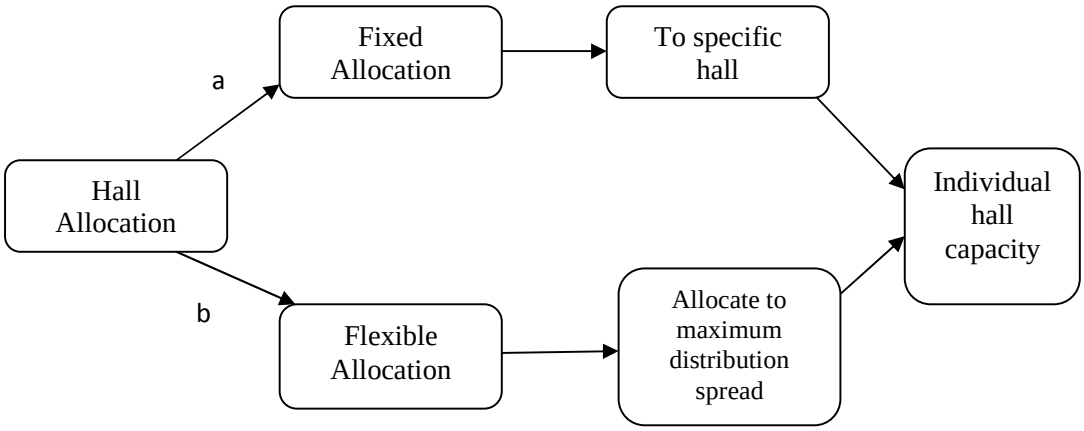

Figure 3.3: Description of Hall Stage Allocation Process. Source: (Adewumi & Ali, 2010)

Following the model given in the CA stage with similar assumptions, we formulate this stage as follows:

If we take x_{ij} to represent the number of students of category i to be allocated to the j-th hall, h_j as the capacity of hall j, p_i as a proportion of the number of students of i-th category in the halls, m as the total number of categories, x_i as used in the CA stage, n as the total number of halls, k as the number of fixed categories in HA allocation and N_i number of halls that have some students of i-th category. The objective of maximizing the spread of students across the halls is modelled as:

$$\max \sum_{i=1}^{m} p_i N_i \tag{3.7}$$

Selections of the objective function respond to weak restrictions of spreading. To illustrate this category k is taken, N_i is the number of halls with at least one student of category k. For this reason, maximizing N_i will maximize the number of halls with students of category k. The students of this category will be spread out more. $p_i = \max_i \{x_i\} - (\max_j \{x_{ij}\} - \min_j \{x_{ij}\})$ is the proportion of students in the halls with fewer students and the number of students in the most populated halls. Maximizing this value will enhance more uniform distribution of the students.

Subject to:

$$\sum_{i=1}^{m} x_{ij} = h_j \tag{3.8}$$

This constraint guarantees that students allocated in each hall are equal to the hall's capacity.

$$\sum_{j=1}^{n} x_{ij} = x_i \tag{3.9}$$

This guarantees that students allocated to each category are equal to CA results.

$$x_{ij} = c_i \text{ for some } j_{(i)} \text{ fixed for each } i = 1...k \tag{3.10}$$

$$x_{ij} = 0 \text{ for other } j \neq j_{(i)} \text{ for each } i = 1...k \tag{3.11}$$

This constraint ensures the students in categories like sports and health are allocated in designated halls as stated by the hard constraint. Meanwhile, p_i is the proportion of students in the i-th category allocated to each hall divided by all students in the i-th category while p_i is the mean of the maximum and the minimum of these rates. With the function to maximize given as $\sum_{i=1}^{m} p_i N_i$, with N_i ensuring that the objective function increases with the quantity of halls with students of category i. In addition, p_i will ensure that students are proportionately distributed to each category. Once a feasible solution is determined, the exact number of students to be allocated from each category per hall will be known. Using this solution, we can allocate students at the floor level.

3.2.3 Floor Allocation Stage

The results from the HA stage serve as input to this stage. This stage determines exactly how many students of each category will be allocated into each floor of each hall. FA is determined separately for each hall. Thereafter, the students who make up these numbers will be distributed across all the floors of the halls in ways that are fair and evenly spread. The fixed categories here are the Ht students who are to be allocated to the lowest floor in their specified halls while the Fy students are to be in the highest possible floor in their halls. Other categories are regarded as

flexible categories and can be allocated to any floor in a way that is evenly spread. The solution is determined exactly as in the HA stage, except that in this case the distribution is determined separately for each hall. For each hall, if a fixed category has been assigned to it, the students will be allocated to the best floors possible and sometimes based on specific needs as in the Ht category (See appendix A). This stage is also modelled as a form of multiple KSP as illustrated in Figure 3.4.

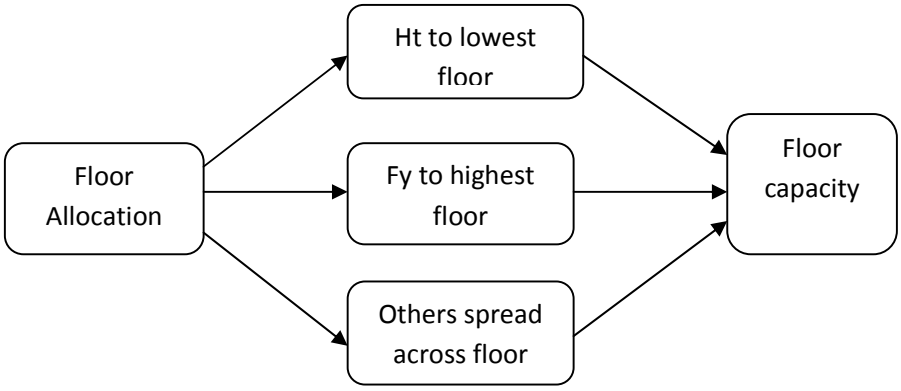

Figure 3.4: Description of Floor Stage Allocation Process. Source: (Adewumi & Ali, 2010)

Following the assumptions made in the previous stage modelling, one can proceed as follows:

The objective is to

$$\max \sum_{i=1}^{m} \sum_{k=1}^{b_j} \sum_{l=1}^{f_j} w_{ijl} \, p_{ijkl} \, N_{ij} \tag{3.12}$$

subject to:

$$\sum_{i=1}^{m} x_{ijlk} = a_{jlk} \tag{3.13}$$

$$\sum_{k=1}^{b_j} \sum_{l=1}^{f_j} x_{ijkl} = x_{ij} \tag{3.14}$$

This constraint guarantees that the number of students allocated from each category in each hall corresponds to the HA results.

where

x_{ijlk} represents the number of students to be allocated of thei-th category in the l-th floor of the k-th block of the j-th hall.

x_{ij} number of students of category i allocated to the j-th hall (result of hall allocation)

b_j number of blocks in the j-th hall.

f_j number of floors in the j-th hall.

a_{jkl} capacity of the l-th floor of the k-th block of the j-th hall.

m total number of categories.

p_{ijkl} proportion (rate) of students of the i-th category in each floor of each block, computed as in p_i for the hall allocation.

w_{ijl} weight of allocating a student of i-th category to the l-th floor of the j-th hall. Weights are assigned in the same way as in Adewumi & Ali (2010) as follows: where the hall has Ht allocation and no Fy allocation: $w_1 = 1$; $w_2 = 0$ with subscript 1 representing the Ht category and subscript 2 representing the Fy category. Where the halls have Fy allocation and no Ht, $w_1 = 0$; $w_2 = 1$ are assigned; and where the hall has both Ht and Fy allocations, $w_1 = 0.7$; $w_2 = 0.3$. The rest of the categories were assigned w = 0.5.

N_{ij} the sum for each block of the number of floors that have some students of i-th category in the j-th Hall.

Since the objective is maximization, the higher the fitness value (set as the objective function), the better the solution provided when the function is evaluated.

3.3. Summary

This chapter was focused on the HSAP. It uses specific examples where the process of space allocation is demonstrated. It showed that the process has several stages that deal with numerous constraints. Mathematical models for all the three stages for the optimisation of the HSAP under the different constraints were developed.

CHAPTER 4

HEURISTICS FOR HOSTEL SPACE ALLOCATION PROBLEM

4.1 Introduction

The challenge involved in obtaining the solution for the analytical models at different stages of the allocation process for the HSAP reflects the computational complexities involved in determining feasible solutions to the problem. As a variant of KSP and an instance of the SAP, an NP-hard problem requires good solution techniques especially as the solution space increases. Burke &Varley (1998) recommend the adoption of heuristics in tackling this type of COP. In this chapter, the techniques and the solution methodology adopted namely the HC, SA, TS, LAHC and GA including hybrids with GA and LAHC are discussed. The purpose of the hybridization is to synergize the strengths of the underlying algorithms for possible improved performance. To enable comparisons to be made, exact solutions using the CPLEX solver incorporated into AIMMS® software were computed and reported.

4.2 Methodology

This section presents an overview of the algorithm of techniques adopted for the HSAP including the exact solution obtained from AIMMS®. The descriptions of each algorithm are given below:

4.2.1 Exact Method

AIMMS® is optimisation modelling software utilised for solving large-scale scheduling and optimisation problems. It comprises of algebraic modelling language, an Integrated Development

Environment (IDE) for model editing as well as Graphical User Interface (GUI) for model viewing and development. AIMMS® incorporated many solvers including CPLEX, KNITRO, SNOPT, IPOPT, and Conopt through the aid of AIMMS open solver interface. The software provides both imperative and declarative programming styles. Optimisation model formulation occurs through declarative language elements like set and indices, as scalar and parameters in multidimensional approach, constraints and variables that are peculiar to all algebraic modelling languages, and permits for a precise description of some problems in the mathematical optimisation domain. The language also supports units of measurements as well as compiles and runtime analysis of unit, which detect modelling errors if employed.

AIMMS supports control flow and procedure statements for data exchange with external data sources like databases, spreadsheet, test files and extensible mark-up language (XML) optimisation models for post and pre-processing tasks, handling of user interface events and the development of hybrid algorithms for some types of problem to which solvers cannot efficiently proffer solutions. AIMMS supports reusability as users can arrange models in libraries of user models for later use. A free version of the software is available online. Figure 4.1 gives a description of the model explorer of the software which is very useful in solving mathematical optimisation problems in various forms including linear, quadratic, nonlinear, mixed-integer, mixed-integer nonlinear, global optimisation, stochastic, robust optimisation and constraint programming problems.

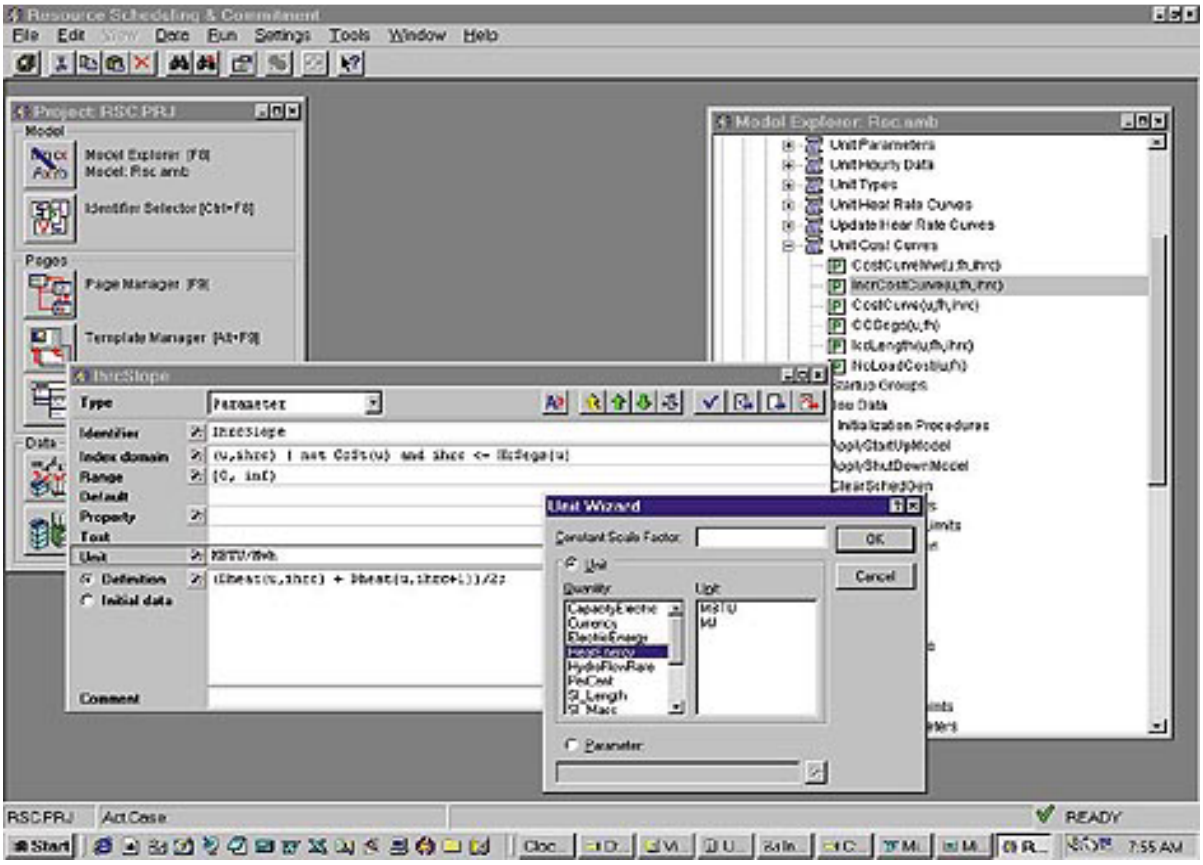

Figure 4.1: Model explorer with parameter form

4.2.2 Genetic Algorithms

GA was first proposed independently by Fraser (1957) and Bremermann (1962). However, Holland (1975) is often cited as the main pioneer research work in the GA field. GA extracts inspiration from the biological concept of the survival of the fittest or the natural selection principle. A population of solutions evolve from one generation to another through a successive combination of a number of operations of selection, crossover and mutation (Goldberg, 1989; Forrest, 1993; Michalewicz, 1996). A solution (individual) is usually encoded as a string (called a chromosome) with several of this forming a population. A new population is generated by copying some fitter individuals from the current population and selecting some newly created

individuals using genetic operators, such as mutation and crossover. Once the termination criteria are met, the algorithms stop. The encoding of the solution into chromosome is an important step that influences the efficiency of GA (Falkenauer, 1997). Usually, the encoding (genotype) should have a one-to-one mapping with the actual phenotype (actual solution).

The crossover operator permits chromosomes to inherit some promising traits from two (possibly more) selected parents while mutation seeks to introduce some new traits that can enhance the solution obtained from the crossover operation (Davis, 1987; Goldberg, 1989). Selection of parents are done via various means including the well known roullete wheel selections. While it is agreed that fitter individuals should have a larger probability of being selected for the new generation, it is also important to permit a few "less-fit" individuals to increase the diversity of the population.

GA has proved to be very efficient in solving many real-life optimisation problems including the HSAP as shown in the pioneer research mentioned earlier. Based on the new model introduced, this research study implemented GA in order to facilitate the comparison of results obtained with other techniques for the HSAP. More information on GA and it variants can be found in literature (examples: Sastry et al., 2005; Goldberg, 1989; Davis, 1991; Beasley et al., 1993; Reeves, 1995;Mitchell, 1996; Michalewicz & Fogel, 2000). An overview of GA as implemented in this work is given in Figure 4.2. Since the models used are based on KSP, a string representation of chromosomes that treat the number of students to be distributed as items, each with associated profit and weight values, to be packed into the knapsack which is the capacity constraints are chosen, depending on the level of allocation.

```
1. Generate an initial random population = best
2. Evaluate the fitness of best = x
3. for each generation i till n do
        3.1. for m iterations to create a new population do
                3.1.1. Select parents
                3.1.2. if random[0,1] ≤ crossover_rate then
                        3.1.2.1. Perform crossover to create new children
                        3.1.2.2. for each child do
                                3.1.2.2.1. if random[0,1] ≤ mutation_rate then
                                        3.1.2.2.1.1. Perform mutation
                                3.1.2.2.2. end if
                        3.1.2.3. end for
                        3.1.2.4. Add children to population P(i)
                3.1.3. else
                        3.1.3.1. Add parents to population P(i)
        3.2. end for
        3.4. Evaluate the fitness of P(i) = x*
        3.5. if x* > x then
                3.5.1. x = x*
                3.5.2. best = P(i)
        3.6. end if
        3.7. if i == n then
                3.7.1. Finished
        3.8. else
                3.8.1.  Goto 3.
        3.9. end if
4. end for
```

Figure 4.2: Genetic Algorithm

4.2.3 Hill Climbing Algorithm

HC is a LS technique that seeks to improve on a current solution by iteratively replacing it with the best solution found within the neighbourhood of the LS space. This algorithm thus continuously moves in the direction of the path that provides a better solution. The main problem with HC is premature convergence, that is, getting stuck in a local optimum. Figure 4.3 shows

the description of the algorithm implemented in this research study. This research attempts to adapt HC to escape the local optimal point by way of hybridization as will be described in later sections.

```
1. Generate an initial random solution = best
2. Evaluate the fitness of best = x
3. for each iteration i till n do
        3.1. candidate = Randomly_Generated_Neighbour N(i)
        3.2. Evaluate fitness of N(i) = x*
        3.3. if x* > x then
                3.3.1. best = candidate
                3.3.2. x = x*
        3.4. end if
        3.5. if i == n then
                3.5.1. Finish
        3.6. else
                3.6.1. Goto 3.
        3.7. end if
4. end for
```

Figure 4.3: The Hill Climbing Algorithm implemented

4.2.4 Tabu Search Algorithm

TS was initially proposed by Fred Glover (Glover, 1977) and was later popularized in Glover (1989) and Glover, (1990). It is a single solution approach that has found a variety of applications in practice. TS makes use of historical information and a memory (tabu list) to prevent the search from cycling and becoming trapped in a local optimum. The tabu list is a short-term memory of recent neighbourhood moves that are prohibited during the search to prevent it from going back to recently visited points in the search space. The length of the tabu list determines how many moves are stored in the list while tabu tenure defines how many iterations of each move in the tabu list are taboos. Although the tabu list is helpful, sometimes it

may restrict the search excessively. Therefore, most TS algorithms have integrated a mechanism called the aspiration criteria, which is used to mitigate the strength of the tabu list. Some long-term memories that store records of the entire search process for the purpose of intensification and diversification are also used. A simple intensification and diversification method can be carried out by introducing incentive or penalty values to modify the evaluation of moves according to the frequency memory (Glover & Laguna, 1995). Other diversification methods are also provided in Soriano & Gendreau, (1996). Details on the TS can be found in Glover &Laguna (1997) and Gendreau (2003).

TS approaches have been used in many practical areas including transportation and routing, scheduling, bioinformatics, telecommunications, network design and graph partitioning and colouring (see Widmer & Hertz, 1989; Reeves, 1995; Skorin-Kapov & Vakharia, 1993; Taillard, 1994; Gendreau et al., 1994; Mazzola & Schantz, 1995; Rolland et al., 1996; Glover & Laguna, 1997). The TS algorithm as implemented in this research work is presented in Figure 4.4.

---

1. Generate an initial random solution = best
2. Set current = working = best
3. Evaluate the fitness of best = f_best
4. Set the fitness of current (f_current) and the fitness of working (f_working) = f_best
5. Initiate the Tabu List TL
6. for each iteration i till n do
    - 6.1. working = Generate_Working(current)
    - 6.2. Evaluate fitness of working = f_working
    - 6.3. if (f_working better then f_current and !Find_Taboo(working)) or (f_working better than f_best) then
        - 6.3.1. iff_working better than f_best then
            - 6.3.1.1. f_best = f_working
            - 6.3.1.2. best = working
        - 6.3.2. end if
        - 6.3.3. Update TL with working
        - 6.3.4. current = working

6.3.5. f_current = f_working
6.4. else
6.4.1. working = current
6.4.2. f_working = f_current
6.5. end if
6.6. if i > n then
6.6.1. Finish
6.7. else
6.7.1. Goto 6
6.8. end if
7. end for

Figure 4.4: The Tabu Search Algorithm

4.2.5 Simulated Annealing Algorithm

SA is a LS method inspired by the physical cooling process of metals (Metropolis et al., 1953). Since its introduction as an optimisation tool by Kirkpatrick et al. (1983), it has been very usefulness and has shown efficiency in handling optimisation problems including graph partitioning and colouring, route-planning, layout design, sequencing and scheduling, timetabling and signal processing(see Carnevali et al., 1985; Sechen et al., 1988; Johnson et al., 1989; Ogbu & Smith, 1990; Abramson, 1991; Johnson et al., 1991; Thompson & Dowsland, 1998; Burke & Kendall, 1999; Tian et al., 1999; Liu, 1999; Chen & Luk, 1999; Bouleimen & Lecocq, 2003). Details on the algorithms and other applications can be found in Dowsland (1995) and Henderson et al. (2003).

SA follows a simple process similar to the HC but has a probability of accepting worse solutions. For example, in a maximisation problem with objective function f and neighbourhood structure N, SA starts from an initial solution and repeatedly generates and transfers to a neighbouring current solution. During this process, SA has the possibility of visiting worse neighbours in order

to escape from a local optima solution. Particularly, a parameter known as temperature *t*, is used to direct the likelihood of moving to worse neighbour solutions. In each iteration, the algorithm accepts all uphill moves (a move which increases the objective value for a maximisation problem) and some of the downhill moves (a decrease in the objective value for a maximisation problem) based on the metropolis probability defined as exp (δ / t) where δ is the variation in the objective function between the new candidate solution and the current solution. A simulated annealing algorithm implemented in this research work is described in Figure 4.5 below.

---

1. Generate an initial random solution = best
2. Set current = working = best
3. Evaluate the fitness of best = f_best
4. Set the fitness of current (f_current) and the fitness of working (f_working) = f_best
5. Initiate starting temperature T and final temperature F
6. while T ≥ F do
    - 6.1. for each iteration i till n do
        - 6.1.1. working = Randomly_Generate_Solution
        - 6.1.2. Evaluate fitness of working = f_working
        - 6.1.3. iff_working better then f_current then
            - 6.1.3.1. use_solution = true
        - 6.1.4. else
            - 6.1.4.1. Calculate acceptance probability P
            - 6.1.4.2. if P > random[0,1] then
                - 6.1.4.2.1. use_solution = true
            - 6.1.4.3. end if
        - 6.1.5. end if
        - 6.1.6. ifuse_solution then
            - 6.1.6.1. use_solution = false
            - 6.1.6.2. f_current = f_working
            - 6.1.6.3. current = working
            - 6.1.6.4. iff_current better then f_best then
                - 6.1.6.4.1. best = current
                - 6.1.6.4.2. f_best = f_current
            - 6.1.6.5. end if
        - 6.1.7. else
            - 6.1.7.1. f_working = f_current
            - 6.1.7.2. working = current
        - 6.1.8. end if
        - 6.1.9. if i > n then

6.1.9.1. end for
6.1.10. else
6.1.10.1. Goto 6.1.
6.1.11. end if
6.2. end for
6.3. Update T according to cooling schedule
6.4. if T < F then
6.4.1. Finish
6.5. else
6.5.1. Goto 6.
6.6. end if
7. end while

Figure 4.5: The Simulated Annealing Algorithm

4.2.5 Late Acceptance Hill Climbing Algorithm

The LAHC was introduced and used by Verstichel & Berghe (2009). Similar to other single-solution search techniques such as the HC, SA and TS, the LAHC starts with a randomly generated initial solution and at each iteration it evaluates a new candidate to determine whether to accept or reject it (Verstichel & Berghe, 2009). In order to apply its acceptance rule, LAHC maintains a list (of a fixed length) of previous values of the current cost function. The candidate cost is compared with the last element of the list and if it is not worse, it is accepted. After the acceptance procedure, the cost of the new current solution is inserted into the beginning of the list and the last element is removed from the end of the list (Abuhamdah, 2010). The inserted current cost is equal to the candidate's cost in cases of acceptance, while in cases of rejection it equals the previous value. The LAHC is memory based consistent with the TS (Taillard, et al, 2001). However, the TS and the LAHC lists have a different nature and purpose. In TS, solutions (or moves) are memorized while in the LAHC the list contains the values of the cost function. Moreover, at each iteration in TS, the candidate solutions are compared with the complete list

whereas in LAHC only one value from the end of the list is used (Edmund & Yuri, 2012). These alterations in the memory utilization mechanism make LAHC less time consuming than TS. Besides, it is possible to make the processing time of LAHC genuinely independent of the length of the list by eliminating the shifting of the whole list at each iteration.

An improvement on the initial idea of the LAHC through the use of "virtual" shifting of the list has been proposed by Edmund & Yuri (2012). However, the list elements are immobile and the list appears as a fitness array F_a of length s_2 its virtual beginning v, at the i^{th} iteration, is calculated as:

$$v = i \bmod L_{fa}$$

Where "mod" represents the remainder from the integer division. At each iteration, the value of f_v is compared with the candidate cost and after accepting or rejecting the current cost, a new value is assigned to f_v.The length L_{fa} appears as a single genuine input parameter for this algorithm. No other parameter is required.

LAHC performance is not affected by the initial values of fitness array (Ozcan et al, 2009). At the beginning of the search, the initial list can contain any arbitrary values. If these are much higher than the initial cost, then the algorithm will generate a corresponding number (equal to the L_{fa}) of random perturbations while filling the list with current costs. If all elements of the initial fitness array are very low, then the algorithm will generate the same number of non-accepted moves and again, will fill the fitness array with the value of the initial cost. However, a very small delay in the search procedure can cause either of the two variants to occur. If one does not

wish to wait until the algorithm does it automatically, then it is possible to set up all elements of the fitness array to be equal to the initial cost before starting the search.

It should be noted that the LAHC uses a greedy acceptance rule process (rejects all worse candidates) only in the case of the delayed comparison (Hu, Kahng, & Tsao, 1995). Nevertheless, if a current solution is accepted with its immediate candidate, LAHC (in a similar way to SA, threshold accepting (TA) and generalized deterministic annealing **(**GDA)) allows the acceptance of worsening moves. This can happen in a situation where the current cost is better than the value from the list and the candidate cost is located amongst them. Considering that accepting worsening moves usually increases the strength of a search process, it can be estimated that the LAHC has a better performance than the greedy HC. Alternatively, there are possible cases where the current cost is worse than the value from the list (Abuhamdah, 2010). Here (using the initial approach of LAHC), a non-worsening move can still be discarded. Such algorithmic behaviour is usually regarded as undesirable in computational search (SA, TA, and GDA always accept non-worsening moves). In order to be persistent with this practice, Burke & Bykov (2008) proposed a second improvement on the initial idea, so that not only "late acceptance" rules can be used for the worsening moves, but also all non-worsening ones can be accepted. Results of the initial experiments in the current research work further show certain advantages of both improvements on the initial idea. All experiments in this research were carried out with the final (improved) version of LAHC. Consequently, its final acceptance condition at the i^{th} iteration is expressed as:

$$c_i^* \leq c_{i-Lfa} \quad \text{or} \quad c_i^* \leq c_{i-1}$$

In this formula, c_i^* is the candidate cost, c_{i-1} is the current cost and c_{i-Lfa} denotes the cost of the current solution L_{fa} iterations, which is equal to $f_{(i \bmod Lfa).}$. Obviously, when L_{fa} is equal to 1 or 0, LAHC is simply greedy HC. Consequently, LAHC achieves its exclusive properties when L_{fa} is equal to two or higher. The algorithm for LAHC is given in Figure 4.6.

1. Generate an initial random solution = best s
2. Evaluate the initial cost function of best s = C(s)
3. Give the length L_{fa}
4. For all iteration $k \in \{0...Lfa-1\}$ $f_k := c(s)$
    4.1. Counter I=0;
    4.2. Do until the end of the condition
        4.2.1. Generate a candidate solution s*
        4.2.2. Evaluate the cost function C(s*) and compute
        4.2.3. $v = i \bmod L_{fa}$
        4.2.4. If $c(s^*) \le f_v$ or $c(s^*) \le c(s)$
            4.2.4.1. Then the candidate is accepted (s:=s*)
        4.2.5. Else, the candidate is rejected (s:=s)
            4.2.5.1. Add current cost into the fitness array **fv**:=**C**(**s**)
    4.3. Update the counter I:=I+1
5. End for

Figure 4.6: Late Acceptance Hill Climbing

4.2.6Hybridization of techniques

Various hybrids of the underlying techniques were attempted in order to enhance the strength of these techniques in the search for better solutions to the HSAP. The algorithms for some of these hybrids are presented in this section.

First, hybrids of LAHC with GA, HC, TS and SA were implemented in this study. For the LAHC_GA hybrid, some GA operators (e.g. crossover and mutation) were introduced in order to enhance the performance of the LAHC technique. Similarly, the GA and LAHC were hybridised

with HC, SA and TS to find solutions to the HSAP at the three stages of the allocation process.

The algorithm for LAHC_GA is given in Figure 4.7.

1. Generate an initial random solution = best s
2. Evaluate the initial cost function of best s = C(s)
3. Give the length L_{fa}
4. For all iteration $k \in \{0...Lfa-1\}$ $f_k := c(s)$
    4.1. counter = 0
    4.2. while counter ≤ length of candidate do
        4.2.1. Select individuals P(counter) and P(counter+1) from candidate
        4.2.2. if random[0,1] ≤ crossover_rate then
            4.2.2.1. Perform crossover
        4.2.3. end if
            4.2.3.1. if random[0,1] ≤ mutation_rate then
                4.2.3.1.1. Perform mutation
        4.2.4. end if
        4.2.5. Increment counter by 2
        4.2.6. if counter > length of candidate then
            4.2.6.1. end while
        4.2.7. else
            4.2.7.1. Goto 3.3.4.
        4.2.8. end if
    4.3 end while
    4.4. Evaluate fitness of candidate = x*
    4.5. if x* > x then
        4.5.1. best = candidate
        4.5.2. x = x*
    4.6. end if
5. Do until the end of the condition
    5.1. Generate a candidate solution s*
    5.2. Evaluate the cost function C(s*) and compute
    5.3. $v = i \bmod L_{fa}$
    5.4. If $c(s^*) \le f_v$ or $c(s^*) \le c(s)$
        5.4.1. Then the candidate is accepted (s:=s*)
        5.4.2. Else the candidate is rejected (s:=s)
    5.5. Add current cost into the fitness array **fv**:=**C**(**s**)
    5.6. Update the counter I:=I+1
6. End for

Figure 4.7.LAHC and GA Hybrid (LAHC_GA)

For the GA hybrids, each individual chromosome represents the students (gene) that have been allocated to each category. These are the students that have been allocated to each category, for each hall, at the HA stage and the students that have been allocated to each category, for each floor of each hall, at the FA stages. The uniform crossover technique was used while mutation was performed by randomly swapping one gene from one individual to the next.

Furthermore, HC was also hybridized with GA (HC_GA) and LAHC (HC_LAHC) in order to explore the LS ability of HC to improve the performance of GA and LAHC. This is done with the hope to provide improved solutions. The algorithms for HC_GA and HC_LAHC are given in Figures 4.8 and 4.9 respectively.

---

1. Generate an initial random solution = best
2. Evaluate the fitness of best = x
3. for each iteration i till n do
    - 3.1. candidate = Randomly_Generated_Neighbour N(i)
    - 3.2. Evaluate fitness of candidate = x*
    - 3.3. if x* > x then
        - 3.3.1. best = candidate
        - 3.3.2. x = x*
        - 3.3.3. counter = 0
        - 3.3.4. while counter ≤ length of candidate do
            - 3.3.4.1. Select individuals P(counter) and P(counter+1) from candidate
            - 3.3.4.2. if random[0,1] ≤ crossover_rate then
                - 3.3.4.2.1. Perform crossover
            - 3.3.4.3. end if
                - 3.3.4.4. if random[0,1] ≤ mutation_rate then
                - 3.3.4.4.1. Perform mutation
            - 3.3.4.5. end if
            - 3.3.4.6. Increment counter by 2
            - 3.3.4.7. if counter > length of candidate then
                - 3.3.4.7.1. end while
            - 3.3.4.8. else
                - 3.3.4.8.1. Goto 3.3.4.
            - 3.3.4.9. end if

3.3.5. end while
3.3.6. Evaluate fitness of candidate = x*
3.3.7. if x* > x then
3.3.7.1. best = candidate
3.3.7.2. x = x*
3.3.8. end if
3.4. end if
3.5. if i == n then
3.5.1. Finish
3.6. else
3.6.1. Goto 3.
3.7. end if
4. end for

Figure 4.8: Hybrid of HC and GA (HC_GA) Algorithm

1. Generate an initial random solution = best
2. Evaluate the fitness of best = x
3. for each iteration i till n do
3.1. candidate = Randomly_Generated_Neighbour N(i)
3.2. Evaluate fitness of candidate = x*
3.3. if x* > x then
3.3.1. best = candidate
3.3.2. x = x*
3.3.3. Evaluate the initial cost function of best s = C(s)
3.3.4. Give the length L_{fa}
3.3.5. For all iteration $k \in \{0...Lfa-1\}$ $f_k := c(s)$
3.3.5.1. Counter I=0;
3.3.5.2. Do until the end of the condition
3.3.5.2.1. Generate a candidate solution s*
3.3.5.2.2. Evaluate the cost function C(s*) and compute
3.3.5.2.3 $v = i \bmod L_{fa}$
3.3.5.3. If $c(s^*) \le f_v$ or $c(s^*) \le c(s)$
3.3.5.3.1. Then the candidate is accepted (s:=s*)
3.3.5.4. Else the candidate is rejected (s:=s)
3.3.5.4.1. Add current cost into the fitness array **fv**:=**C**(**s**)
3.3.5.4.2. Update the counter I:=I+1
3.3.6. end for
3.7. end if
3.8. if i == n then
3.8.1. Finish

3.9. else
3.6.1. Goto 3.
3.10. end if
4. end for

Figure 4.9: Hybrid of HC and LAHC (HC_LAHC) Algorithm

SA was also hybridized with GA (SA_GA) and LAHC (SA_LAHC) in order to explore the ability of SA to accept worse solutions compared with GA and LAHC techniques. The SA_GA and SA_LAHC algorithms are given in Figures 4.10 and 4.11 respectively.

1. Generate an initial random solution = best
2. Set current = working = best
3. Evaluate the fitness of best = f_best
4. Set the fitness of current (f_current) and the fitness of working (f_working) = f_best
5. Initiate starting temperature T and final temperature F
6. while T ≥ F do
    6.1. for each iteration i till n do
        6.1.1. working = Randomly_Generate_Solution
        6.1.2. Evaluate fitness of working = f_working
        6.1.3. iff_working better then f_current then
            6.1.3.1. use_solution = true
        6.1.4. else
            6.1.4.1. Calculate acceptance probability P
            6.1.4.2. if P > random[0,1] then
                6.1.4.2.1. use_solution = true
            6.1.4.3. end if
        6.1.5. end if
        6.1.6. ifuse_solution then
            6.1.6.1. use_solution = false
            6.1.6.2. f_current = f_working
            6.1.6.3. current = working
            6.1.6.4. counter = 0
            6.1.6.5. while counter ≤ length of working do
                6.1.6.5.1. Select individuals P(counter) and P(counter+1) from working
                6.1.6.5.2. if random[0,1] ≤ crossover_rate then

6.1.6.5.2.1. Perform crossover
6.1.6.5.3. end if
6.1.6.5.4. if random[0,1] ≤ mutation_rate then
6.1.6.5.4.1. Perform mutation
6.1.6.5.5. end if
6.1.6.5.6. Increment counter by 2
6.1.6.5.7. if counter > length of working then
6.1.6.5.7.1. end while
6.1.6.5.8. else
6.1.6.5.8.1. Goto 6.1.6.5.
6.1.6.5.9. end if
6.1.6.6. end while
6.1.6.7. Evaluate fitness of working = f_working
6.1.6.7. iff_working better then f_current then
6.1.6.7.1. f_current = f_working
6.1.6.7.2. current = working
6.1.6.8. iff_current better then f_best then
6.1.6.8.1. best = current
6.1.6.8.2. f_best = f_current
6.1.6.9. end if
6.1.7. else
6.1.7.1. f_working = f_current
6.1.7.2. working = current
6.1.8. end if
6.1.9. if i > n then
6.1.9.1. end for
6.1.10. else
6.1.10.1. Goto 6.1.
6.1.11. end if
6.2. end for
6.3. Update T according to cooling schedule
6.4. if T < F then
6.4.1. Finish
6.5. else
6.5.1. Goto 6.
6.6. end if
7. end while

Figure 4.10: Hybrid of SA with GA (SA_GA) Algorithm

1. Generate an initial random solution = best
2. Set current = working = best
3. Evaluate the fitness of best = f_best
4. Set the fitness of current (f_current) and the fitness of working (f_working) = f_best
5. Initiate starting temperature T and final temperature F

6. while T ≥ F do
    6.1. for each iteration i till n do
        6.1.1. working = Randomly_Generate_Solution
        6.1.2. Evaluate fitness of working = f_working
        6.1.3. iff_working better then f_current then
            6.1.3.1. use_solution = true
        6.1.4. else
            6.1.4.1. Calculate acceptance probability P
            6.1.4.2. if P > random[0,1] then
                6.1.4.2.1. use_solution = true
            6.1.4.3. end if
        6.1.5. end if
        6.1.6. ifuse_solution then
            6.1.6.1. use_solution = false
            6.1.6.2. f_current = f_working
            6.1.6.3. current = working
            6.1.6.4. Evaluate the initial cost function of best s = C(s)
            6.1.6.5. Give the length L_{fa}
            6.1.6.6. For all iteration $k \in \{0...Lfa-1\}$ $f_k := c(s)$
                6.1.6.6.1. Counter I=0;
                6.1.6.6.2. Do until the end of the condition
                    6.1.6.6.2.1. Generate a candidate solution s*
                    6.1.6.6.2.2. Evaluate the cost function C(s*) and compute
                    6.1.6.6.2.3 $v = i \bmod L_{fa}$
                6.1.6.6.3. If $c(s^*) \le f_v$ or $c(s^*) \le c(s)$
                    6.1.6.6..3.1. Then the candidate is accepted (s:=s*)
                6.1.6.6.4. Else the candidate is rejected (s:=s)
                    6.1.6.6.4.1. Add current cost into the fitness array **fv**:=**C(s)**
                    6.1.6.6.4.2. Update the counter I:=I+1
            6.1.6.7. end for
        6.1.8. else
            6.1.8.1. f_working = f_current
            6.1.8.2. working = current
        6.1.9. end if
        6.1.10. if i > n then
            6.1.10.1. end for
        6.1.11. else
            6.1.11.1. Goto 6.1.
        6.1.12. end if
    6.2. end for
    6.3. Update T according to cooling schedule
    6.4. if T < F then
        6.4.1. Finish
    6.5. else
        6.5.1. Goto 6.

6.6. end if
7. end while

Figure 4.11: Hybrid of SA with LAHC (SA_LAHC) Algorithm

Similarly, TS was hybridized with GA (TS_GA) and LAHC (TS_LAHC) in order to synergize the strengths of these algorithms. The algorithms for TS_GA and TS_LAHC are given in Figures 4.12 and 4.13 respectively.

1. Generate an initial random solution = best
2. Set current = working = best
3. Evaluate the fitness of best = f_best
4. Set the fitness of current (f_current) and the fitness of working (f_working) = f_best
5. Initiate the Tabu List TL
6. for each iteration i till n do
    6.1. working = Generate_Working(current)
    6.2. Evaluate fitness of working = f_working
    6.3. iff_working better then f_current and !Find_Taboo(working) or f_working better thenf_best then
        6.3.1. iff_working better then f_best then
            6.3.1.1. f_best = f_working
            6.3.1.2. best = working
        6.3.2. end if
        6.3.3. Update TL with working
        6.3.4. current = working
        6.3.5. f_current = f_working
        6.3.6. counter = 0
        6.3.7. while counter ≤ length of working do
            6.3.7.1. Select individuals P(counter) and P(counter+1) from working
            6.3.7.2. if random[0,1] ≤ crossover_rate then
                6.3.7.2.1. Perform crossover
            6.3.7.3. end if
            6.3.7.4. if random[0,1] ≤ mutation_rate then
                6.3.7.4.1. Perform mutation
            6.3.7.5. end if
            6.3.7.6. Increment counter by 2
            6.3.7.7. if counter > length of working then
                6.3.7.7.1. end while
            6.3.7.8. else
                6.3.7.8.1. Goto 6.3.7.
            6.3.7.9. end if
        6.3.8. end while
        6.3.9. Evaluate fitness of working = f_working

6.3.10. iff_working better then f_current and !Find_Taboo(working) or
f_working better then f_best then
6.3.10.1. iff_working better then f_best then
6.3.10.1.1. f_best = f_working
6.3.10.1.2. best = working
6.3.10.2. end if
6.3.10.3. Update TL with working
6.3.10.4. current = working
6.3.10.5. f_current = f_working
6.3.11. end if
6.4. else
6.4.1. working = current
6.4.2. f_working = f_current
6.5. end if
6.6. if i > n then
6.6.1. Finish
6.7. else
6.7.1. Goto 6
6.8. end if
7. end for

Figure 4.12: Hybrid of TS with GA (TS_GA)algorithm

1. Generate an initial random solution = best
2. Set current = working = best
3. Evaluate the fitness of best = f_best
4. Set the fitness of current (f_current) and the fitness of working (f_working) = f_best
5. Initiate the Tabu List TL
6. for each iteration i till n do
6.1. working = Generate_Working(current)
6.2. Evaluate fitness of working = f_working
6.3. iff_working better then f_current and !Find_Taboo(working) or f_working better thenf_best then
6.3.1. iff_working better then f_best then
6.3.1.1. f_best = f_working
6.3.1.2. best = working
6.3.2. end if
6.3.3. Update TL with working
6.3.4. current = working
6.3.5. f_current = f_working

6.3.6. Evaluate the initial cost function of best s = C(s)
6.3.7. Give the length L_{fa}
6.3.8. For all iteration $k \in \{0...Lfa-1\}$ $f_k := c(s)$
6.3.8.1. Counter I=0;
6.3.8.2. Do until the end of the condition
6.3.8.2.1. Generate a candidate solution s*
6.3.8.2.2. Evaluate the cost function C(s*) and compute

6.3.8.2.3. $v = i \bmod L_{fa}$

6.3.8.3. If $c(s^*) \leq f_v$ or $c(s^*) \leq c(s)$

6.3.8.3.1. Then the candidate is accepted (s:=s*)

6.3.8.4. Else the candidate is rejected (s:=s)

6.3.8.4.1. Add current cost into the fitness array **fv**:=**C**(**s**)

6.3.8.4.2. Update the counter I:=I+1

6.3.8.5. end for

6.3.10.3. Update TL with working

6.3.10.4. current = working

6.3.10.5. f_current = f_working

6.3.11. end if

6.4. else

6.4.1. working = current

6.4.2. f_working = f_current

6.5. end if

6.6. if i > n then

6.6.1. Finish

6.7. else

6.7.1. Goto 6

6.8. end if

7. end for

Figure 4.13: Hybrid of TS with LAHC (TS_LAHC) algorithm

In the next chapter, details of the simulation experiments conducted are presented along with the settings and the results obtained for these techniques based on the three data sets used for the HSAP as stated previously.

CHAPTER 5

EXPERIMENTAL SETTINGS AND RESULTS

5.1 Introduction

As stated earlier, the allocation into male and female hostels follows the same but mutually exclusive procedure. Thus, male and female student populations are considered separately in the implementation and allocation processes. Some of the results obtained for the male and female allocations are therefore presented separately at each stage of the allocation process. Furthermore, the performances of the heuristic algorithms are compared by evaluating their fitness values. The algorithms with the best fitness values will represent the heuristics that have provided the best solutions for each stage. For example, at the HA and FA stages, the performance of the heuristics depend on the distribution (spread across halls or floors for the flexible categories) obtained from each algorithm. Results obtained are essentially the distribution (not the actual physical allocation), that is, the exact number of students from each category that are to be accommodated at each stage of the allocation process.

5.2 Parameter Settings

Most heuristics have parameters that determine their behaviours and performance. Wrong settings and/or combination of parameters can cause a good technique to performance badly in its search for solutions to a problem. It is thus important to find good parameter settings for algorithms before their execution. Often times, this can be obtained from previous research work

and available literature. However, it is pertinent to perform some simulation experiments on this problem to determine the best parameter settings.

Parameters can be kept static (fixed) throughout the execution of their underlying technique or varied dynamically according to some pre-defined criteria or testing procedure. This study adopted the static approach as the initial parameter setting used for each of the algorithms which remain the same at each stage of the allocation process. A series of simulation experiments were performed using the real dataset (data set one) to find optimal parameter combinations that give the best results consistently. Values of various parameters were varied for different runs and the result obtained compared to determine the best value. This was done for all the implemented algorithms. The parameter combination and values that gave the highest fitness (in terms of constraint satisfaction) and the best students' distributions were chosen and later used in the actual experiment.

At the end of the processing for each algorithm, the parameters obtained are described in this section. Some of the parameters were tuned according to the size of the problem instance and by observing many runs of simulation experiments. Based on this, the number of iterations for HC, TS, SA, and LAHC was set to 200. Similarly, the number of iterations for GA was set to 200. A tabu list of size10 is specified for TS while LAHC has a tabu list size of 20. The initial temperature for SA was set to 80 while the final temperature was set to 0.5. The temperature decreases by an alpha of 0.99. These settings remain the same for GA and LAHC hybrids. GA has a population size of 20, a crossover rate of 0.7 and a mutation rate of 0.1. Each algorithm was executed 100 times with each problem instance or dataset and the best results in terms of solution quality are recorded. The simulation experiment was performed on a stand-alone

desktop computer with an Intel dual core processor at 1.86 GHz and 2GB RAM, running on Windows 7 Ultimate operating system. Coding was done using MATLAB 7.10.

5.3 Testing

Figure 5.1represents the general framework (block diagram) for testing all the stages of the HSAP. The pseudo-code associated with this block diagram is given below (Figure 5.2).

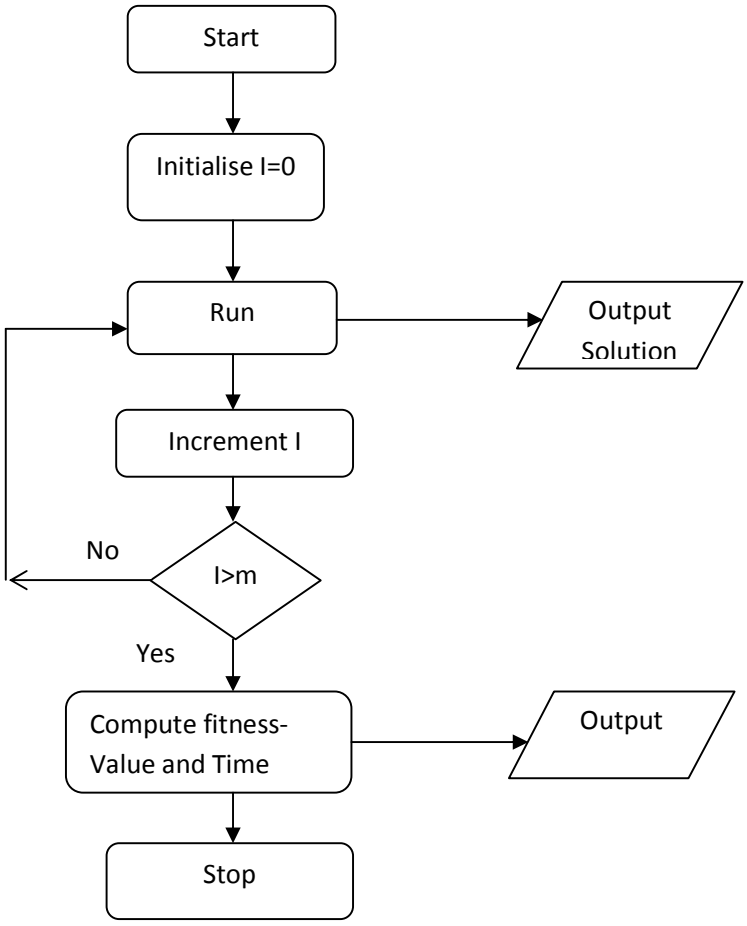

Figure 5.1: Framework for algorithm testing

1. Select data set
2. Open log files
3. Set m, max_it, good_profit (, temperature if using SA or a SA variant)
4. Initialize inner_time, temp, eval_func, cnt
5. For i from 1 to m
    1. Select algorithm
    2. Run algorithm
    3. Write to Type 1 log file
    4. if the current run was stopped by the good_profit condition
        1. cnt=cnt+1;
    5. endif
    6. Compute inner_time, temp, eval_func
6. End
7. Compute Fitness_Values, Time
8. Write to Type 2 log file

Figure 5.2: The pseudocode for the testing method

A detailed description of Figure 5.2 is given below:

1. Select data set:

    – in this step the data sets are stored in the variables that are used as inputs for the algorithm used;

    – the input variables for the MATLAB function (implementations of the algorithms) which depend on the data set currently used are:

        - CA:
            - total_cap: total capacity of the hostels
            - cat_app: the applicants classified in categories
        - HA:
            - H: hall capacities

        - S: determines whether the HA stage is solved for male or female students
        - ca: the results from the category allocation stage (the distribution of students from each category)
    - FA:
        - H: hall capacities
        - S: determines whether the HA stage is solved for male or female
        - fa: the current hall number

2. Open log files:
    - in a bit to simplify the means by which the results are saved, two types of log files were designed:
        - Type 1 log files – used to store the solution given at each program run
        - Type 2 log files – used to store the other measures of performance of the algorithms
3. Set m, max_it, good_profit and temperature
    - Given the structure of the testing method, additional variables are used:
        - m – the number of program runs
        - max_it – the maximum number of iterations (the first stopping condition for every algorithm)
        - good_profit – the objective value threshold at which it is considered that the algorithm has reached convergence; this represents the second stopping condition; the values were determined by first running the algorithms max_it

times, taking the values close to the steady state values of the objective function and computing them as follows:

$$good\_profit = \max(eval\_func) - 5\%(\max(eval\_func) - \min(eval\_func))$$

where eval_func is the mean value of the objective function in each program run.

- temperature – is the initial temperature used for SA and SA hybrids.

4. Initialize inner_time, temp, eval_func, cnt.

   – Another set of temporary variables used are needed to compute the performance measures defined earlier. The roles of these variables are defined as follows:

   - inner_time– is the vector of length *m*which stores the computation time of the currently selected algorithm at each program run.
   - temp– is a m x max_itmatrix that stores all the transient values of the objective function for each program run; it is used to compute an average transient sequence of the objective function to illustrate the convergence of the currently selected algorithm.
   - eval_func– is a vector of *m* elements that stores the best objective value returned by the algorithm for each program run.
   - cnt– is a variable that counts how many of the program runs are stopped by the good_profit condition, meaning how many program runs converge in less than max_ititerations.

5. For i from 1 to *m* {run the algorithm *m* times}

- Select algorithm – based on the string stored in alg variable the program will switch between the algorithms; the values that the alg variable can take are: “HC”, ”GA”, ”HC_GA”, ”HC_LAHC”, ”LAHC”, “LAHC_GA”, “SA”, “SA_GA”, “SA_LAHC”, “TS”, “TS_GA”, “TS_LAHC”.

- Run algorithm – in this step the selected algorithm is run with the configurations made in the previous steps.
- Write to Type 1 log file – in the log file the solution will be written; the number of solutions will be *m*.
- Check the stopping condition
- Store the values needed for computing the performance measures in the corresponding temporary variables.

6. Compute the performance measures – Fitness Values and Times

7. Write to the Type 2 log file – the performance measures computed at the previous step will be written in the log files.

5.4 Results and Discussions

Simulation experiments were done using the three datasets. At the CA stage, the Ht, Sp and Fo categories are considered fixed categories for male and female students. At the HA stage, the Ht, Sc and Sp categories are fixed and are to be assigned to the 1^{st}, 2^{nd} and 5^{th} halls respectively (for male students) and 1^{st}, 3^{rd} and 6^{th} halls respectively (for female students) as specified in the requirements and in Appendix A. The result obtained at the CA stage is deterministic (AIMMS) and only results obtained for one execution of the program are shown. For comparison purposes, the results of the separate runs are shown in this thesis with the results obtained from AIMMS. It should be noted that the AIMMS software could not obtain any exact solution at the HA and FA stages as the number of constraints become too large to handle. This further justifies the use of

heuristics at all the three stages. The results of AIMMS, evidently the best (see Table 5.7), was computed at the CA stage to serve as the initial solution and input for the HA and FA stages.

After testing all algorithms at the CA stage, the solution obtained by the best algorithms become the input to the HA stage. Similarly, the best solution obtained by the best algorithms at the HA stage is used as the input for the FA stages. Performance comparisons can be done by viewing the exact number of students each algorithm assigns to accommodation, and their fitness values; the value of the fitness function or objective function for an individual is its score. Algorithm performances were compared with each other by evaluating their ability to determine improved solutions over a specific number of iterations. The execution times of the algorithms were also noted.

Results are presented in tabular form for the HA and FA stages to show the numbers of students from each category that are allocated to each hall and into each floor with regards to the relevant technique. The effectiveness of the algorithms was determined by comparing the most feasible solutions found by each algorithm over a specific number of iterations. For the FA stages, results for the halls are shown.

5.4.1 CA Stage Results

Tables 5.1-5.6 show the number of students from each category obtained from the various techniques at this stage of the allocation process. The result is benchmarked with the exact solution (distribution) obtained with the AIMMS software. The tables represent the results for all three data sets and for both male and female students.

The results of the heuristic techniques show similar patterns to that of the AIMMS software. All results satisfied the constraints associated with this stage and have similar students' distribution with that obtained from the AIMMS. All fixed category students (Fo, Ht and Sp) were allocated by the use of all the techniques. The performance of the heuristics was determined by observing their ability to generate feasible solutions over 200 iterations (a measure of their fitness values) and comparing them with that of AIMMS. At this stage of the allocation process, the higher the fitness value, the better the solution. Tables 5.7-5.9 show the performances of the algorithms for the three data sets and for male and female students. The best fitness values and the shortest times determined by the algorithms are emphasized (in bold) in the tables presented below for the purpose of clarity.

Data Set One

Table 5.1: Category Allocation for male students

Category	**Fo**	**Ht**	**Sp**	**Fy**	**Sc**	**Fr**	**Ds**	**Ot**	**Total**
Applicant	20	70	400	1240	400	1332	100	1800	5362
Category Results by Each Algorithm									
AIMMS	20	70	400	1240	400	1332	100	348	3910
HC	20	70	400	1240	400	1332	100	348	3910
TS	20	70	400	1240	400	1332	100	348	3910
SA	20	70	400	1164	400	1188	61	607	3910
LAHC	20	70	400	1240	400	1332	100	348	3910
GA	20	70	400	920	160	1277	30	1033	3910
LAHC_HC	20	70	400	1240	400	1332	100	348	3910
LAHC_TS	20	70	400	1240	400	1332	100	348	3910
LAHC_SA	20	70	400	1240	400	949	100	731	3910
GA_HC	20	70	400	1240	400	1332	100	348	3910
GA_TS	20	70	400	1240	400	1332	100	348	3910
GA_SA	20	70	400	1240	400	1007	51	722	3910
GA_LAHC	20	70	400	1240	400	1332	100	348	3910

Table 5.2: Category Allocation for female students

Category	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
Applicant	25	80	500	1240	230	1367	60	1000	4502
Category Results by Each Algorithm									
AIMMS	25	80	500	1260	230	1367	53	73	3588
HC	25	80	500	1262	230	1367	53	71	3588
TS	25	80	500	1420	230	641	60	632	3588
SA	25	80	500	1420	118	1367	48	30	3588
LAHC	25	80	500	1313	230	1367	60	13	3588
GA	25	80	500	1420	119	1367	60	17	3588
LAHC_HC	25	80	500	1420	230	1258	13	62	3588
LAHC_TS	25	80	500	1307	230	1367	60	19	3588
LAHC_SA	25	80	500	780	32	1124	60	987	3588
GA_HC	25	80	500	1420	123	1367	60	13	3588
GA_TS	25	80	500	1420	230	1246	45	42	3588
GA_SA	25	80	500	1420	230	1236	60	37	3588
GA_LAHC	25	80	500	1420	230	1270	52	11	3588

Data Set Two

Table 5.3: Category Allocation for male students

Category	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
Applicant	30	105	600	1860	600	1998	150	2700	8043
Category Results by Each Algorithm									
HC	30	105	600	1717	600	1998	66	749	5865
TS	30	105	600	1860	600	1998	150	522	5865
SA	30	105	600	1860	600	1998	150	522	5865
LAHC	30	105	600	1860	600	1998	150	522	5865
GA	30	105	600	1860	600	1811	112	747	5865
LAHC_HC	30	105	600	1860	600	1998	150	522	5865
LAHC_TS	30	105	600	1860	600	1998	150	522	5865
LAHC_SA	30	105	600	1717	600	1998	66	749	5865
GA_HC	30	105	600	1860	600	1998	150	522	5865
GA_TS	30	105	600	1860	600	1998	150	522	5865
GA_SA	30	105	600	1573	600	1998	53	906	5865
GA_LAHC	30	105	600	1860	600	1998	150	522	5865

Table 5.4: Category Allocation for female students

Algorithm	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
Applicant	38	120	750	2130	345	2051	90	1500	7024
Category Results by Each Algorithm									
HC	38	120	750	1851	181	1851	21	570	5382
TS	38	120	750	1512	114	1654	90	1104	5382
SA	38	120	750	1959	345	2051	86	33	5382
LAHC	38	120	750	2130	345	1678	25	296	5382
GA	38	120	750	2130	345	1869	23	107	5382
LAHC_HC	38	120	750	1950	237	2051	73	163	5382
LAHC_TS	38	120	750	2130	345	1320	41	638	5382
LAHC_SA	38	120	750	2130	345	1115	84	800	5382
GA_HC	38	120	750	1933	260	2051	10	220	5382
GA_TS	38	120	750	1681	212	2051	14	516	5382
GA_SA	38	120	750	2130	345	802	74	1123	5382
GA_LAHC	38	120	750	2130	345	1907	40	52	5382

Data Set Three

Table 5.5: Category Allocation for male students

Category	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
Applicant	45	158	900	2790	900	2997	225	4050	12065
Category Results by Each Algorithm									
HC	45	158	900	2790	900	2997	225	783	8798
TS	45	158	900	2790	900	2600	42	1363	8798
SA	45	158	900	2790	900	2997	225	783	8798
LAHC	45	158	900	2790	900	2997	225	783	8798
GA	45	158	900	2790	900	2997	225	783	8798
LAHC_HC	45	158	900	2301	75	2467	124	2728	8798
LAHC_TS	45	158	900	2790	900	2997	225	783	8798
LAHC_SA	45	158	900	2790	900	2997	225	783	8798
GA_HC	45	158	900	2790	900	2997	225	783	8798
GA_TS	45	158	900	2790	900	1395	225	2385	8798
GA_SA	45	158	900	2790	900	2625	160	1220	8798
GA_LAHC	45	158	900	1603	583	2770	36	2703	8798

Table 5.6: CA for female students

Category	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
Applicant	57	180	1125	3195	518	3077	135	2250	10537
Category Results by Each Algorithm									
HC	57	180	1125	3195	518	2197	60	742	8074
TS	57	180	1125	3195	518	2664	128	207	8074
SA	57	180	1125	2949	486	3077	38	162	8074
LAHC	57	180	1125	2698	518	3077	96	323	8074
GA	57	180	1125	3195	518	2409	56	534	8074
LAHC_HC	57	180	1125	3195	518	2030	135	834	8074
LAHC_TS	57	180	1125	2574	518	3077	78	465	8074
LAHC_SA	57	180	1125	3195	518	2409	56	534	8074
GA_HC	57	180	1125	2919	97	3077	88	531	8074
GA_TS	57	180	1125	2774	518	3077	126	217	8074
GA_SA	57	180	1125	2917	160	2092	135	1408	8074
GA_LAHC	57	180	1125	3195	518	2092	97	810	8074

Data Set One

Table 5.7: Performance of the algorithms at CA stage (Data Set 1)

	Male		Female	
Algorithms	**Fitness Values**	**Time (Minutes)**	**Fitness Values**	**Time (Minutes)**
AIMMS	**2086.726**	15	**2108.888**	10
HC	2086.725	**6**	2108.824	**7**
TS	2079.626	7	2100.884	**6**
SA	2080.636	8	2104.828	7
LAHC	2085.526	7	2107.624	7
GA	2082.726	44	2105.554	43
LAHC_HC	2085.824	476	2101.804	402
LAHC_TS	2085.726	378	2103.894	386
LAHC_SA	2086.626	113	2106.424	120
GA_HC	2083.726	43	2099.624	46
GA_TS	2082.936	43	2105.924	40
GA_SA	2084.796	42	2104.874	42
GA_LAHC	2085.826	21	2107.824	24

Data Set Two

Table 5.8: Performance of the algorithms at CA stage (Data Set 2)

	Male		Female	
Algorithms	**Fitness Values**	**Time (Minutes)**	**Fitness Values**	**Time (Minutes)**
HC	2443.203	**23**	2420.286	25
TS	2443.427	24	2419.87	26
SA	2443.507	**23**	2421.416	**21**
LAHC	2443.793	**23**	2421.309	22
GA	2445.559	53	2422.748	41
LAHC_HC	2446.422	374	2422.248	291
LAHC_TS	2446.322	364	**2422.778**	309
LAHC_SA	**2446.522**	334	2422.648	305
GA_HC	2445.559	45	2422.548	43
GA_TS	2446.422	51	2422.748	48
GA_SA	2446.134	43	2422.738	41
GA_LAHC	2445.846	52	2422.604	50

Data Set Three

Table 5.9: Performance of the algorithms at CA stage (Data set 3)

	Male		Female	
Algorithms	**Fitness Values**	**Time (Minutes)**	**Fitness Values**	**Time (Minutes)**
HC	**2509.812**	28	2300.059	22
TS	2475.200	26	2249.362	27
SA	2472.476	**24**	2282.819	**18**
LAHC	2472.997	35	2266.284	25
GA	2506.759	30	2306.88	29
LAHC_HC	2512.101	200	2304.256	194
LAHC_TS	2525.855	216	2299.713	200
LAHC_SA	2508.044	197	**2312.229**	184
GA_HC	2502.551	32	2288.774	49
GA_TS	2487.988	53	2293.353	40
GA_SA	2481.246	37	2286.362	40
GA_LAHC	2497.256	37	2308.004	25

Tables 5.7, 5.8 and 5.9 show the performances of all algorithms for the three data sets for both male and female students. From the results presented on Table 5.7 from data set one, AIMMS was able to find the best value (data set one) compared to other algorithms for male and female student distributions. Additionally, HC and TS determined near optimal solutions for male and female students in the shortest time duration (6 minutes). The best fitness value for data set two and three are recorded from LAHC_SA and HC respectively for male students' allocation. Also for the male students' results, HC, SA and LAHC had the shortest possible times of 23 minutes and 24 minutes for data sets two and three respectively. However, for data sets two and three for female students, LAHC_TS and LAHC_SA have the best fitness values of 2422.778 and 2312.229 respectively. SA had the shortest times of 21 and 18 minutes for female allocations for data sets two and three respectively. The results of AIMMS, LAHC_SA and HC are used as the inputs for the HA stage for both male and female allocations since they generated the best fitness values in both cases.

5.4.2 HA Stage Results

Once the solutions are received from the CA stage, the number of students from each category that will be allocated into specific halls is determined. These allocations are shown in Tables 5.10-5.15 for data sets one, two and three respectively (for both male and female students) while the comparative performance of the algorithms are presented in Tables 5.16-5.18 for the three data sets respectively.

Table 5.10: Hall Allocation for ale students (Data Set 1)

Hall	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
1	5	70	0	240	0	259	19	67	660
2	1	0	0	9	400	13	11	10	444
3	5	0	0	328	0	355	21	91	800
4	1	0	0	409	0	431	22	105	968
5	5	0	400	46	0	50	11	14	526
6	3	0	0	208	0	224	16	61	512
Total	20	70	400	1240	400	1332	100	348	3910

Table 5.11: Hall Allocation for female students (Data Set 1)

	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
H1	7	80	0	252	230	273	10	14	866
H2	7	0	0	349	0	375	14	19	764
H3	2	0	0	125	0	137	5	7	276
H4	4	0	0	238	0	260	9	13	524
H5	5	0	0	298	0	317	10	16	646
H6	0	0	500	0	0	5	5	2	512
Total	25	80	500	1262	230	1367	53	71	3588

Table 5.12: Hall Allocation for male students (Data Set 2)

Hall	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
H1	3	105	0	333	0	392	12	145	990
H2	5	0	0	22	600	24	5	10	666
H3	7	0	0	451	0	528	17	197	1200
H4	4	0	0	551	0	646	13	238	1452
H5	6	0	600	71	0	72	8	32	789
H6	5	0	0	289	0	336	11	127	768
Total	30	105	600	1717	600	1998	66	749	5865

Table 5.13: Hall Allocation for female students (Data Set 2)

Hall	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
H1	4	120	0	431	345	264	8	127	1299
H2	6	0	0	591	0	363	11	175	1146
H3	8	0	0	208	0	131	4	63	414
H4	7	0	0	401	0	251	7	120	786
H5	8	0	0	495	0	306	11	149	969
H6	5	0	750	4	0	5	0	4	768
Total	38	120	750	2130	345	1320	41	638	5382

Table 5.14: Hall Allocation for male students (Data Set 3)

Hall	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
H1	11	158	0	541	0	581	43	151	1485
H2	6	0	0	41	900	38	3	11	999
H3	11	0	0	736	0	788	59	206	1800
H4	9	0	0	888	0	961	71	249	2178
H5	1	0	900	115	0	125	11	32	1184
H6	7	0	0	469	0	504	38	134	1152
Total	45	158	900	2790	900	2997	225	783	8798

Table 5.15: Hall Allocation for female students (Data Set 3)

Hall	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
H1	9	180	0	644	518	482	11	106	1950
H2	10	0	0	886	0	667	10	146	1719
H3	10	0	0	307	0	241	10	53	621
H4	10	0	0	602	0	454	12	101	1179
H5	13	0	0	748	0	560	8	125	1454
H6	5	0	1125	8	0	5	5	3	1151
Total	57	180	1125	3195	518	2409	56	534	8074

Table 5.16: Performance of the algorithms at HA stage (Data Set 1)

	Male		Female	
Algorithms	**Fitness Values**	**Time (Minutes)**	**Fitness Value**	**Time (Minutes)**
HC	111708	**3**	124826	**3**
TS	111701	4	124825	7
SA	111709	15	129489	5
LAHC	111710	5	124824	4
GA	111728	6	129492	8
LAHC_HC	111729	15	129494	15
LAHC_TS	111728	15	129493	13
LAHC_SA	111727	4	129491	7
GA_HC	111730	5	129490	8
GA_TS	111726	5	129494	6
GA_SA	111731	6	**129495**	9
GA_LAHC	**111732**	10	129493	12

Table 5.17: Performance of the algorithms at HA stage (Data Set 2)

	Male		Female	
Algorithms	**Fitness Values**	**Time (Minutes)**	**Fitness Values**	**Time (Minutes)**
HC	92530	48	92049	50
TS	92531	**46**	92044	47
SA	92532	47	92048	**33**
LAHC	88670	47	92046	46
GA	92524	474	92175	470
LAHC_HC	88671	99	92045	93
LAHC_TS	88669	100	92047	96
LAHC_SA	92525	63	92050	54
GA_HC	92526	4162	92195	4233
GA_TS	88668	3864	92150	4139
GA_SA	**92533**	4247	**92225**	4344
GA_LAHC	92527	590	92043	578

Table 5.18: Performance of the algorithms at HA stage (Data Set 3)

	Male		Female	
Algorithms	**Fitness Values**	**Time (Minutes)**	**Fitness Values**	**Time (Minutes)**
HC	113334	54	102011	53
TS	113333	64	102014	44
SA	113332	**37**	102010	**36**
LAHC	113326	46	102013	41
GA	113417	507	102017	464
LAHC_HC	113327	91	102012	119
LAHC_TS	113325	117	102015	116
LAHC_SA	113328	63	102009	67
GA_HC	113447	4630	102016	4659
GA_TS	**113448**	4029	102017	3686
GA_SA	113424	5371	**102018**	5080
GA_LAHC	113418	616	102014	630

As stated earlier, the total number of students, from each category, that are allocated into each hall must not exceed the capacity of the hall. Also, the accumulated number of students from the same category that have been allocated across the halls must be equivalent to the knapsack of allocated students as determined by the best heuristic at the CA stage. Also, the allocation of the flexible categories of students must be done in proportion to their priority factors from the CA. These hard constraints do not have to be satisfied for a feasible solution. As can be ascertained from Tables 5.10 - 5.15, these constraints are satisfied by all the algorithms with feasible allocation results achieved. For example, Ht, Sp and Sc are allocated to halls one, five and two respectively as required for male students' distributions, while other categories of students were evenly distributed across other halls (compared with constraints given in appendix A for specified halls for each category).

In the flexible category, students need to be distributed throughout the halls as evenly spread as possible. Based on the assumption made and modelling at this stage of the allocation process, the higher the fitness values the better the distribution. Results for this are presented in Tables 5.16, 5.17 and 5.18.As can be established from Table 5.16, GA_LAHC and GA_SA clearly have the highest fitness values for male and female students respectively while HC has overall, the shortest time for data set one as extracted from this Table. All other heuristic algorithms' performances are similar to this.

For data set two (Table 5.17), GA_SA has the best fitness value while TS and SA have the shortest times in execution compared to other algorithms. Data set 3 (Table 5.18) produced GA_TS and GA_SA with the best fitness values for male and female student distributions respectively. GA_SA showed consistent strength for all three datasets.

Generally, the hybrid algorithms performed better than the individual heuristic algorithms. In view of the fact that GA_LAHC/GA_SA, GA_TS and GA_SA have the best performance for data sets one, two and three respectively, and for male and female students in terms of fitness values, these solutions are used as the inputs for the FA stage.

5.4.3 Floor Allocation Stage Results

FA stage results are determined separately for each hall. They are generated using the results of the best heuristic from the HA stage for individual hall solutions as input. The distribution of students to the floors of each hall is done exactly the same way as in the HA stage. The only difference is that the distribution is to the floors and blocks of each hall and not to the halls

directly. Also, similar to the HA stage, the results of the algorithms are compared to determine which algorithm gives the best distributions. Furthermore, a comparison of the execution times for the algorithms for each hall allocation is computed and reported.

Fixed categories assigned to specific halls were treated first as being assigned to specific but sufficient number of floors while the flexible categories were distributed by the algorithms to fill up the remaining bed spaces in a way that maximizes the spread of the categories. Tables 5.19–5.36 show these results (the exact number of male and female students for each category)as allocated to each floor. The results were similar for the three datasets, showing consistent behaviour regardless of the size of the data, hence only the results of dataset one are reported in this thesis. However, a comparative performance of the algorithms in providing feasible solutions (based on their fitness values and execution time) for all the three data sets are presented. This is based on their performance in terms of even distribution of students across the floors of each hall with regards to the specified constraints. Similar to the HA stage, the higher the fitness values, the better the distribution.

Data Set One

The experimental results obtained for data set one are presented below for male and female allocations. In the comparative performance, the algorithm with the best fitness value and shortest execution time is shown (in bold for clarity) in each table of results.

Hall One

A study of Table 5.19, in relation to the given constraints, shows that results obtained satisfy as many constraints as possible. Only Ht category students are allocated to this hall among the fixed

categories for the FA stage. Furthermore, as many as possible of the Ht students are allocated to the lowest floor. Block 1 floor 1 represents ground floor, therefore, in this research study, a greater percentage of students being allocated to the ground level is avoided. The flexible category students fill up the remaining spaces.

Table 5.19: Male allocation throughout the floors of hall 1

Category	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
Block 1 Floor 1	3	1	0	6	0	7	0	1	18
Block 1 Floor 2	0	12	0	36	0	39	3	10	100
Block 1 Floor 3	0	10	0	38	0	39	3	10	100
Block 1 Floor 4	0	12	0	36	0	39	3	10	100
Block 1 Floor 5	0	1	0	4	0	6	0	1	12
Block 2 Floor 1	0	1	0	6	0	10	0	1	18
Block 2 Floor 2	0	10	0	38	0	39	3	10	100
Block 2 Floor 3	0	10	0	38	0	39	0	13	100
Block 2 Floor 4	0	10	0	34	0	39	7	10	100
Block 2 Floor 5	2	3	0	4	0	2	0	1	12
Total	5	70	0	240	0	259	19	67	660

Table 5.20 shows the allocation of female students to hall 1. Ht and Sc are allocated to this hall and to no other fixed category. The distributions of other categories are done in a similar way to that of the male allocation. The comparative results of the algorithms for both male and female students in hall 1 are given in Table 5.21

Table 5.20: Female allocation across the floor of Hall 1

Category	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
Block 1 Floor 1	0	0	0	0	0	0	0	0	0
Block 1 Floor 2	0	1	0	19	0	10	0	0	30
Block 1 Floor 3	0	1	0	7	0	10	0	12	30
Block 2 Floor 1	0	1	0	19	10	10	0	0	40
Block 2 Floor 2	0	2	0	12	10	16	0	0	40
Block 2 Floor 3	0	10	0	17	10	3	0	0	40
Block 3 floor 1	0	12	0	17	10	1	0	0	40
Block 3 floor 2	0	5	0	15	10	10	0	0	40
Block 3 Floor 3	0	3	0	11	10	16	0	0	40
Block 4 Floor 1	0	0	0	0	0	0	0	0	0
Block 4 Floor 2	0	5	0	7	20	28	0	0	60
Block 4 Floor 3	0	5	0	7	20	28	0	0	60
Block 4 Floor 4	0	5	0	13	20	22	0	0	60
Block 5 Floor 1	0	5	0	17	19	18	0	1	60
Block 5 Floor 2	0	5	0	17	15	18	5	0	60
Block 5 Floor 3	0	5	0	17	15	18	5	0	60
Block 5 Floor 4	0	5	0	17	15	23	0	0	60
Block 6 Floor 1	0	0	0	0	0	0	0	0	0
Block 6 Floor 2	0	2	0	8	10	9	0	1	30
Block 6 Floor 3	3	2	0	8	10	7	0	0	30
Block 7 Floor 1	2	3	0	12	13	13	0	0	43
Block 8 Floor 1	2	3	0	12	13	13	0	0	43
Total	7	80	0	252	230	273	10	14	866

Table 5.21: Comparison of the performance of the algorithms for Hall 1

Halls	Algorithms	Male		Female	
		Fitness Values	**Time (Minutes)**	**Fitness Values**	**Time (Minutes)**
Hall 1	HC	615540	**13**	15251875	22
	TS	615540	15	15519017	25
	SA	720425	16	16137809	24
	LAHC	615540	14	15303373	**21**
	GA	307944	118	8858633	131
	LAHC_HC	618160	27	15813022	28
	LAHC_TS	618160	37	16052314	29
	LAHC_SA	**879532**	26	16165621	27
	GA_HC	802068	391	17357262	337
	GA_TS	798067	411	**20340692**	356
	GA_SA	811128	138	18295324	323
	GA_LAHC	618160	80	6937668	105

The comparisons show that LAHC_SA and GA_TS are clearly the best performers for the male and female distributions respectively. The other algorithms perform very similarly to this, especially the hybrids ones, this is also a feature observed when compared to the pattern in the HA allocation stage. HC and LAHC have the shortest execution times for male and female distributions respectively.

Hall Two

As can be ascertained from Table 5.22, all the hard constraints are satisfied in allocating the categories of male students to each floor also as the Sc students are allocated to hall 2 as required. From Table 5.23, female students are evenly distributed across all blocks and floor levels. No Ht or Sp students are allocated to hall 2.This is permitted as they must be allocated to halls 1 and 6 respectively.

Table 5.22: Male Allocation throughout the floor of Hall 1

Category	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
Block 1 Floor 1	0	0	0	0	0	0	0	0	0
Block 1 Floor 2	0	0	0	0	40	0	0	0	40
Block 1 Floor 3	0	0	0	0	40	0	0	0	40
Block 1 Floor 4	0	0	0	0	40	0	0	0	40
Block 2 Floor 1	0	0	0	0	0	0	0	0	0
Block 2 Floor 2	0	0	0	1	39	0	0	0	40
Block 2 Floor 3	0	0	0	0	40	0	0	0	40
Block 2 Floor 4	0	0	0	2	35	2	1	0	40
Block 3 Floor 1	0	0	0	0	0	0	0	0	0
Block 3 Floor 2	0	0	0	0	38	2	0	0	40
Block 3 Floor 3	0	0	0	1	28	2	9	0	40
Block 3 Floor 4	0	0	0	0	37	2	1	0	40
Block 4 Floor 1	0	0	0	0	0	0	0	0	0
Block 4 Floor 2	0	0	0	2	25	1	0	0	28
Block 4 Floor 3	0	0	0	2	24	2	0	0	28
Block 4 Floor 4	1	0	0	1	14	2	0	10	28
Total	1	0	0	9	400	13	11	10	444

Table 5.23: Female Allocation throughout the floor of Hall 2

Category	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
Block 1 Floor 1	0	0	0	0	0	0	0	0	0
Block 1 Floor 2	0	0	0	32	0	0	0	0	32
Block 1 Floor 3	0	0	0	32	0	0	0	0	32
Block 1 Floor 4	0	0	0	32	0	0	0	0	32
Block 2 Floor 1	0	0	0	23	0	21	0	4	48
Block 2 Floor 2	0	0	0	37	0	39	0	4	80
Block 2 Floor 3	0	0	0	25	0	51	0	4	80
Block 2 Floor 4	0	0	0	25	0	51	2	2	80
Block 3 Floor 1	0	0	0	9	0	9	2	0	20
Block 3 Floor 2	0	0	0	20	0	56	3	1	80
Block 3 Floor 3	0	0	0	25	0	51	3	1	80
Block 3 Floor 4	0	0	0	36	0	39	4	1	80
Block 4 Floor 1	0	0	0	18	0	20	0	2	40
Block 4 Floor 2	4	0	0	17	0	19	0	0	40
Block 4 Floor 3	3	0	0	18	0	19	0	0	40
Total	7	0	0	349	0	375	14	19	764

A comparison of the performances of each algorithm as extracted from Table 5.24 shows that LAHC_TS clearly performs best for male and female distributions. The other algorithms have similar performances, especially the hybrids ones. LAHC and HC have the shortest execution times for male and female students respectively.

Table 5.24: Comparison of the Performance of the algorithms for Hall 2

Halls	Algorithms	Male		Female	
		Fitness Values	**Time (Minutes)**	**Fitness Values**	**Time (Minutes)**
Hall 2	HC	195256	27	6159741	**19**
	TS	202032	25	5807991	21
	SA	202024	31	5617424	22
	LAHC	214516	**19**	5881764	20
	GA	43406	95	2055892	106
	LAHC_HC	507005	42	5989451	25
	LAHC_TS	**1145336**	58	**13236300**	31
	LAHC_SA	483523	40	5748126	25
	GA_HC	375599	530	6203093	964
	GA_TS	373086	652	6191173	1292
	GA_SA	496112	396	7043637	532
	GA_LAHC	213598	49	3484308	83

Hall Three

Table 5.25 shows how the male students are allocated to hall 3. No fixed categories are allocated to this hall. Fitness value comparisons are shown in Table 5.27. From the results, GA_TS and LAHC_TS have the best fitness values for male and female distributions. In addition, HC and LAHC have the shortest execution times compare to other algorithms. Similarly, Table 5.26

shows the female distributions across hall 3. The hard constraints are satisfied and the total distribution of students does not exceed the capacity of this hall.

Table 5.25: Male Allocation throughout the floor of Hall 3

	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
Block 1 Floor 1	0	0	0	8	0	8	0	4	20
Block 1 Floor 2	0	0	0	25	0	29	0	6	60
Block 1 Floor 3	0	0	0	25	0	29	0	6	60
Block 1 Floor 4	0	0	0	25	0	29	0	6	60
Block 2 Floor 1	0	0	0	8	0	8	0	4	20
Block 2 Floor 2	0	0	0	25	0	28	0	7	60
Block 2 Floor 3	0	0	0	25	0	26	0	9	60
Block 2 Floor 4	0	0	0	25	0	26	0	9	60
Block 3 Floor 1	0	0	0	8	0	8	0	4	20
Block 3 Floor 2	0	0	0	25	0	26	3	6	60
Block 3 Floor 3	0	0	0	25	0	26	3	6	60
Block 3 Floor 4	0	0	0	25	0	26	3	6	60
Block 4 Floor 1	0	0	0	8	0	8	2	2	20
Block 4 Floor 2	0	0	0	25	0	26	3	6	60
Block 4 Floor 3	2	0	0	25	0	26	3	4	60
Block 4 Floor 4	3	0	0	21	0	26	4	6	60
Total	5	0	0	328	0	355	21	91	800

Table 5.26: Female Allocation throughout the floor of Hall 3

	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
Block 1 Floor 1	0	0	0	0	0	0	0	0	0
Block 1 Floor 2	0	0	0	10	0	20	0	0	30
Block 1 Floor 3	0	0	0	21	0	9	0	0	30
Block 2 Floor 1	0	0	0	10	0	30	0	0	40
Block 2 Floor 2	0	0	0	18	0	22	0	0	40
Block 2 Floor 3	0	0	0	10	0	29	0	1	40
Block 3 Floor 1	0	0	0	0	0	0	0	0	0
Block 3 Floor 2	0	0	0	18	0	6	0	0	24
Block 3 Floor 3	0	0	0	18	0	6	0	0	24
Block 4 Floor 1	0	0	0	0	0	0	0	0	0
Block 4 Floor 2	0	0	0	11	0	13	0	0	24
Block 4 Floor 3	2	0	0	9	0	2	5	6	24
Total	2	0	0	125	0	137	5	7	276

Table 5.27: Comparison of the performance of the algorithms for Hall 3

Halls	Algorithms	Male		Female	
		Fitness Values	**Time (Minutes)**	**Fitness Values**	**Time (Minutes)**
Hall 3	HC	754544	**17**	943646	18
	TS	754544	21	902381	19
	SA	779682	16	866169	24
	LAHC	754544	17	958086	**17**
	GA	654994	108	243339	105
	LAHC_HC	754544	24	963015	22
	LAHC_TS	754544	40	**1841212**	39
	LAHC_SA	806034	27	920616	21
	GA_HC	**907752**	928	1089711	551
	GA_TS	779104	1106	1205684	769
	GA_SA	849830	455	1300190	907
	GA_LAHC	743294	83	604929	82

Hall Four

Tables 5.28 and 5.29 present male and female distributions respectively across the blocks and floors for hall 4. No fixed categories are allocated to this hall. Fitness value comparisons are shown on Table 5.30. From the results, GA_TS and LAHC_TS have the best fitness values for male and female distributions respectively. GA and LAHC have the shortest execution times compared to other algorithms.

Table 5.28: Male students' allocated throughout the floor of hall 4

	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
Block 1 Floor 1	0	0	0	8	0	8	2	2	20
Block 1 Floor 2	0	0	0	28	0	26	0	6	60
Block 1 Floor 3	0	0	0	28	0	26	0	6	60
Block 1 Floor 4	0	0	0	28	0	26	0	6	60
Block 2 Floor 1	0	0	0	4	0	5	2	1	12
Block 2 Floor 2	0	0	0	28	0	26	0	6	60
Block 2 Floor 3	0	0	0	28	0	26	0	6	60
Block 2 Floor 4	0	0	0	28	0	26	0	6	60
Block 3 Floor 1	0	0	0	5	0	6	0	1	12
Block 3 Floor 2	0	0	0	24	0	29	1	6	60
Block 3 Floor 3	0	0	0	24	0	29	1	6	60
Block 3 Floor 4	0	0	0	24	0	29	1	6	60
Block 4 Floor 1	0	0	0	4	0	7	0	1	12
Block 4 Floor 2	0	0	0	24	0	27	1	8	60
Block 4 Floor 3	0	0	0	24	0	26	1	9	60
Block 4 Floor 4	0	0	0	24	0	26	1	9	60
Block 5 Floor 1	0	0	0	4	0	5	0	3	12
Block 5 Floor 2	0	0	0	24	0	26	5	5	60
Block 5 Floor 3	0	0	0	24	0	26	4	6	60
Block 5 Floor 4	1	0	0	24	0	26	3	6	60
Total	1	0	0	409	0	431	22	105	968

Table 5.29: Female allocation throughout the floor of hall 4

	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
Block 1 Floor 1	0	0	0	27	0	3	0	0	30
Block 1 Floor 2	0	0	0	15	0	12	0	13	40
Block 1 Floor 3	0	0	0	12	0	28	0	0	40
Block 1 Floor 4	0	0	0	18	0	22	0	0	40
Block 2 Floor 1	0	0	0	13	0	17	0	0	30
Block 2 Floor 2	0	0	0	18	0	22	0	0	40
Block 2 Floor 3	0	0	0	18	0	22	0	0	40
Block 2 Floor 4	0	0	0	18	0	22	0	0	40
Block 3 Floor 1	0	0	0	0	0	0	0	0	0
Block 3 Floor 2	0	0	0	16	0	16	0	0	32
Block 3 Floor 3	0	0	0	16	0	16	0	0	32
Block 3 Floor 4	0	0	0	16	0	16	0	0	32
Block 4 Floor 1	0	0	0	16	0	16	0	0	32
Block 4 Floor 2	0	0	0	16	0	16	0	0	32
Block 4 Floor 3	1	0	0	14	0	17	0	0	32
Block 4 Floor 4	3	0	0	5	0	15	9	0	32
Total	4	0	0	238	0	260	9	13	524

Table 5.30: Comparison of the performance of the algorithms for hall 4

Halls	Algorithms	Male		Female	
		Fitness Values	Time (Minutes)	Fitness Values	Time (Minutes)
Hall 4	HC	1126740	23	3747166	19
	TS	1126740	23	3682748	21
	SA	1145042	20	3599103	19
	LAHC	1126740	18	3640744	**17**
	GA	1126662	**5**	1089881	104
	LAHC_HC	1126740	27	3615572	27
	LAHC_TS	1126740	40	**9688018**	33
	LAHC_SA	1234936	28	3559936	25
	GA_HC	1381332	899	4277576	1080
	GA_TS	**1410295**	1051	4032865	1313
	GA_SA	1323007	430	4866551	260
	GA_LAHC	1100514	83	2109015	82

Hall Five

Table 5.31 shows the categories distribution for hall 4 with the Sp category assigned to this hall. Furthermore, Table 5.32 shows the female distributions across all the blocks and floors of this hall with the total distribution not exceeding the given capacity of each hall. Comparisons of the fitness values are shown in Table 5.33. LAHC_TS has the best fitness values for male and female distributions. HC and LAHC have the shortest execution times.

Table 5.31: Male allocation throughout the floor of hall 5

	Fo	**Ht**	**Sp**	**Fy**	**Sc**	**Fr**	**Ds**	**Ot**	**Total**
Block 1 Floor 1	0	0	28	4	0	1	1	6	40
Block 2 Floor 1	0	0	20	3	0	0	10	7	40
Block 3 Floor 1	0	0	32	4	0	4	0	0	40
Block 4 Floor 1	0	0	32	4	0	4	0	0	40
Block 5 Floor 1	0	0	32	4	0	4	0	0	40
Block 6 Floor 1	0	0	32	4	0	4	0	0	40
Block 7 Floor 1	0	0	32	4	0	4	0	0	40
Block 8 Floor 1	0	0	32	3	0	5	0	0	40
Block 9 Floor 1	0	0	32	3	0	5	0	0	40
Block 10 Floor 1	0	0	32	4	0	4	0	0	40
Block 11 Floor 1	0	0	32	3	0	5	0	0	40
Block 12 Floor 1	0	0	32	3	0	5	0	0	40
Block 13 Floor 1	5	0	32	3	0	5	0	1	46
Total	5	0	400	46	0	50	11	14	526

Table 5.32: Male allocation throughout the floors of hall 5

	Fo	**Ht**	**Sp**	**Fy**	**Sc**	**Fr**	**Ds**	**Ot**	**Total**
Block 1 Floor 1	0	0	0	28	0	30	0	2	60
Block 1 Floor 2	0	0	0	30	0	28	0	2	60
Block 1 Floor 3	0	0	0	30	0	29	0	1	60
Block 2 Floor 1	0	0	0	28	0	12	0	0	40
Block 2 Floor 2	0	0	0	28	0	70	0	2	100
Block 2 Floor 3	0	0	0	30	0	68	0	2	100
Block 3 Floor 1	0	0	0	20	0	6	0	0	26
Block 3 Floor 2	0	0	0	28	0	70	0	2	100
Block 3 Floor 3	5	0	0	76	0	4	10	5	100
Total	5	0	0	298	0	317	10	16	646

Table 5.33: Comparison of the performance of the algorithms for Hall 5

Halls	Algorithms	Male		Female	
		Fitness Values	**Time (Minutes)**	**Fitness Values**	**Time (Minutes)**
Hall 5	HC	267796	**21**	2800789	16
	TS	251103	24	2725159	18
	SA	252826	23	2879460	19
	LAHC	255307	22	2698552	**15**
	GA	154843	97	873002	104
	LAHC_HC	262348	23	2714914	20
	LAHC_TS	**698244**	37	**5266183**	31
	LAHC_SA	260912	25	2728788	21
	GA_HC	361283	812	3192745	662
	GA_TS	328913	960	3176483	1043
	GA_SA	385257	726	3499579	1569
	GA_LAHC	109268	93	1812695	82

Hall 6

Table 5.34 shows the categories allocation for hall6 with no fixed categories allocated to the hall. Table 5.35 shows the female distributions across the hall with the fixed Sp students (female) allocated to it. The total distribution of the students did not exceed given capacities. The comparisons of the fitness values are shown in Table 5.36. LAHC_SA and LAHC_TS have the highest fitness values for both male and female distributions respectively and are considered the best solutions. HC and TS have the shortest execution times for male and female distributions respectively.

Table 5.34: Male students' allocation throughout the floor of Hall 6

	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
Block 1 Floor 1	0	0	0	32	0	35	2	11	80
Block 1 Floor 2	0	0	0	58	0	63	6	17	144
Block 1 Floor 3	0	0	0	60	0	63	4	17	144
Block 1 Floor 4	3	0	0	58	0	63	4	16	144
Total	3	0	0	208	0	224	16	61	512

Table 5.35: Female allocation throughout the floor of Hall 6

	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot	Total
Block 1 Floor 1	0	0	80	0	0	0	0	0	80
Block 2 Floor 1	0	0	144	0	0	0	0	0	144
Block 3 Floor 1	0	0	144	0	0	0	0	0	144
Block 4 Floor 1	0	0	132	0	0	5	5	2	144
Total	0	0	500	0	0	5	5	2	512

Table 5.36: Comparison of the performance of the algorithms for Hall 6

Halls	Algorithms	Male		Female	
		Fitness Values	Time (Minutes)	Fitness Values	Time (Minutes)
Hall 6	HC	285440	**5**	124082	12
	TS	285440	9	126121	**8**
	SA	285440	7	125091	21
	LAHC	285440	6	124238	9
	GA	131895	112	9973	90
	LAHC_HC	285440	17	244753	142
	LAHC_TS	285440	34	**280120**	371
	LAHC_SA	**346696**	19	221535	94
	GA_HC	285440	10	235235	487
	GA_TS	285440	13	221535	735
	GA_SA	285440	14	227211	4111
	GA_LAHC	285440	19	145577	58

Data Set Two

The distributions for dataset two are very similar to those of dataset one discussed above hence, the distribution is not presented here due to space limitations. However, the performance of the algorithms for each hall is presented.

The comparisons of the algorithms in terms of their fitness values and execution times for both male and female students in all the halls are given in Tables 5.37–5.42.The algorithm with the highest fitness value is usually the best algorithm (solution). The best fitness values and shortest execution times are put in bold letter in each table for the purpose of clarity. Table 5.37 shows that GA_SA and GA_HC are clearly the best performers for male and female distributions respectively while the other algorithms perform very similarly to the pattern observed in the HA allocation stage. The distributions obtained by the GA_SA and GA_HC are the best solutions in terms of effective distribution of students to the floors of each hall for male and female students respectively. HC has the shortest execution time for male and female student distributions.

Furthermore, Table 5.38 shows the performance of the algorithms at hall 2. LAHC_TS and GA_TS have the best distributions for male and female students respectively. HC and LAHC have the shortest execution times for male and female distributions. Table 5.39 shows clearly the performances of all the implemented algorithms in hall 3 GA_TS and LAHC_TS have the best feasible solutions for male and female students respectively. LAHC has the shortest execution time for both male and female students' distributions. Also, Table 5.40 shows how each of the implemented algorithms performed in allocating students to hall four. From the results, GA_HC and LAHC_TS have the highest fitness values. LAHC has the shortest execution time for both

male and female students' distributions. Table 5.41 gives the detailed performances for all the algorithms for hall five at the FA stage. The results show that LAHC_TS outperformed all other algorithms in distributing male and female students to hall five. LAHC had the shortest execution time for both male and female distributions. Finally, Table 5.42 shows the comparisons of the implemented algorithm for hall six. The results clearly show that GA_SA and LAHC_TS have the highest fitness values for male and female distributions respectively. However, SA and LAHC have the shortest execution times for male and female allocations to hall five.

Table 5.37: Performance of algorithms at FA stage for Hall 1

Hall	Algorithms	Male		Female	
		Fitness Values	**Time (Minutes)**	**Fitness Values**	**Time (Minutes)**
Hall 1	HC	1011454	**23**	4300121	**74**
	TS	1011454	28	4286735	80
	SA	1105322	38	4320852	83
	LAHC	1011454	24	4273418	75
	GA	496580	916	2682923	825
	LAHC_HC	1023914	66	4574601	78
	LAHC_TS	1023914	166	4335068	94
	LAHC_SA	1326652	76	4218129	80
	GA_HC	1340302	2764	**6502593**	824
	GA_TS	1336416	1324	5527632	1506
	GA_SA	**1420966**	773	5780719	1119
	GA_LAHC	1023914	95	2509336	650

Table 5.38: Performance of algorithms at FA stage for Hall 2

Hall	Algorithms	Male		Female	
		Fitness Values	Time (Seconds)	Fitness Values	Time (Minutes)
Hall 2	HC	1529532	**63**	1984862	80
	TS	1543003	89	1835967	86
	SA	1562891	81	1802612	78
	LAHC	1548826	69	1925070	**65**
	GA	298644	760	883516	712
	LAHC_HC	1662805	77	1792910	90
	LAHC_TS	**4028300**	158	2199672	3879
	LAHC_SA	1576627	92	1813910	88
	GA_HC	2104848	2952	2487012	8235
	GA_TS	2806978	1454	**4073679**	159
	GA_SA	2220272	836	2618740	395
	GA_LAHC	1373147	556	1734632	476

Table 5.39: Performance of algorithms at FA stage for Hall 3

Hall	Algorithms	Male		Female	
		Fitness Values	Time (Minutes)	Fitness Values	Time (Minutes)
Hall 3	HC	1159382	54	458923	66
	TS	1159382	63	450669	85
	SA	1172694	82	438875	78
	LAHC	1159382	**40**	453346	**62**
	GA	991956	717	129135	748
	LAHC_HC	1169360	47	472527	87
	LAHC_TS	1169360	179	**817466**	164
	LAHC_SA	1388184	111	458010	89
	GA_HC	1548290	6485	608976	4319
	GA_TS	**1589968**	3728	607531	2076
	GA_SA	1387626	2537	710231	292
	GA_LAHC	1139444	475	448816	508

Table 5.40: Performance of algorithms at FA stage for Hall 4

Hall	Algorithms	Male		Female	
		Fitness Values	Time (Minutes)	Fitness Values	Time (Minutes)
Hall 4	HC	1813307	53	1532111	66
	TS	1813307	72	1551997	75
	SA	1859987	79	1525560	82
	LAHC	1813307	**51**	1596735	**64**
	GA	1737656	738	578502	794
	LAHC_HC	1852250	61	1576401	94
	LAHC_TS	1852250	177	**4473946**	160
	LAHC_SA	2021223	119	1457635	107
	GA_HC	**2376261**	4699	2419639	3858
	GA_TS	2156221	2895	2002689	2101
	GA_SA	2293723	2206	2448671	617
	GA_LAHC	1774418	477	1280123	526

Table 5.41: Performance of algorithms at FA stage for Hall 5

Hall	Algorithms	Male		Female	
		Fitness Values	Time (Minutes)	Fitness Values	Time (Minutes)
Hall 5	HC	595194	89	1262747	80
	TS	627434	99	1299939	79
	SA	612759	89	1277910	74
	LAHC	597103	**78**	1299058	**73**
	GA	454428	715	381824	729
	LAHC_HC	610224	88	1281781	99
	LAHC_TS	**1632083**	186	**3029095**	153
	LAHC_SA	605417	103	1193542	79
	GA_HC	944003	1488	1767901	4643
	GA_TS	1314464	1351	2388121	2521
	GA_SA	1045248	1538	1725143	483
	GA_LAHC	348614	591	1093599	479

Table 5.42: Performance of algorithms at FA stage for Hall 6

Hall	Algorithms	Male		Female	
		Fitness Values	**Time (Minutes)**	**Fitness Values**	**Time (Minutes)**
Hall 6	HC	197792	96	125125	17
	TS	197792	133	125125	15
	SA	213116	**50**	128150	66
	LAHC	197792	92	125125	14
	GA	102201	801	27240	618
	LAHC_HC	197792	140	642864	246
	LAHC_TS	197792	152	**650752**	637
	LAHC_SA	302884	88	499730	215
	GA_HC	472169	720	578498	1331
	GA_TS	473011	1588	632254	2294
	GA_SA	**501468**	360	498237	15722
	GA_LAHC	197792	453	199104	69

Data Set Three

The student distributions for data set three are similar to those of the other data sets described previously; the only exception is that the sizes of the hall capacities differ. The number of halls, floors and blocks for this data set are the same with the others. Therefore, only the performances of the algorithms for each hall are discussed. The comparisons of the algorithms in term of their fitness values and times of execution for both male and female students in all the halls are given in Tables 5.43–5.48. The higher the fitness values, the better the solutions. The best fitness values and shortest execution times are shown in the table of results. The comparisons of the performances in Table 5.43 show that LAHC_TS clearly is the best performer for both male and female distributions. HC and LAHC have the shortest execution time for male and female students' distributions respectively.

Furthermore, Table 5.44 shows the performance of the algorithms for hall two. LAHC_TS has the best distributions for both male and female students. HC and LAHC have the shortest execution times for male and female students' distributions. Table 5.45 shows the performance of all the implemented algorithms in hall three. GA_HC and LAHC_TS have the best feasible solutions for male and female students respectively. LAHC and HC have the shortest execution times for male and female distributions respectively.

Similarly, Table 5.46 shows how each of the implemented algorithms performed in allocating students to hall four. From the results, GA_HC and LAHC_TS have the highest fitness values. LAHC and SA have the shortest execution times for male and female students' distributions. Table 5.47 gives the detailed performances of all algorithms for hall five at the FA stage. The results show that LAHC_TS outperformed all other algorithms in distributing both male and female students to hall five while LAHC has the shortest execution time for both male and female students' distributions.

Table 5.48 shows the comparisons of the implemented algorithms for hall six. The results clearly show that GA_HC and LAHC_TS have the highest fitness values for male and female students' distributions respectively. HC has the shortest execution times for both male and female allocations to this hall.

Table 5.43: Performance of the algorithms at FA stage for Hall 1

Hall	Algorithms	Male		Female	
		Fitness Values	Time (Minutes)	Fitness Values	Time (Minutes)
Hall 1	HC	957623	**10**	13029192	46
	TS	957623	14	11836040	61
	SA	957623	17	15055113	43
	LAHC	957623	18	8998077	**42**
	GA	515185	401	7308776	494
	LAHC_HC	2213417	16	13211108	60
	LAHC_TS	**4800719**	89	15260758	71
	LAHC_SA	1880020	15	11643060	60
	GA_HC	960085	16	15961318	6732
	GA_TS	960085	17	**16051929**	8201
	GA_SA	960085	19	14279887	8358
	GA_LAHC	957623	12	5087450	347

Table 5.44: Performance of the algorithms at FA stage for Hall 2

Hall	Algorithms	Male		Female	
		Fitness Values	Time (Minutes)	Fitness Values	Time (Minutes)
Hall 2	HC	201792	**11**	5662577	**50**
	TS	201792	12	5768102	64
	SA	201792	16	5851004	48
	LAHC	201792	13	5776616	46
	GA	174696	28	1982981	345
	LAHC_HC	1073863	35	6104478	72
	LAHC_TS	**4360212**	79	**14225279**	88
	LAHC_SA	686310	13	6096504	73
	GA_HC	212496	17	7154299	3874
	GA_TS	212496	18	5998834	5297
	GA_SA	212496	23	6765762	4767
	GA_LAHC	201792	33	3383250	258

Table 5.45: Performance of the algorithms at FA stage for Hall 3

Hall	Algorithms	Male		Female	
		Fitness Values	Time (Minutes)	Fitness Values	Time (Minutes)
Hall 3	HC	1714048	50	969516	**42**
	TS	1714048	61	940574	48
	SA	1714048	51	950242	57
	LAHC	1714048	**31**	969516	46
	GA	1598838	384	243339	361
	LAHC_HC	1714048	42	1003975	58
	LAHC_TS	1714048	107	**3260533**	91
	LAHC_SA	1992152	47	939106	61
	GA_HC	**2319483**	4353	1112725	1467
	GA_TS	1714048	47	1051542	2839
	GA_SA	1714048	49	1190947	1838
	GA_LAHC	1714048	42	582226	262

Table 5.46: Performance of the algorithms at FA stage Hall 4

Hall	Algorithms	Male		Female	
		Fitness Values	Time (Minutes)	Fitness values	Time (Minutes)
Hall 4	HC	2974072	47	3389686	53
	TS	2974072	41	3564891	59
	SA	2974072	44	3657268	**52**
	LAHC	2974072	**40**	3436718	55
	GA	3243218	408	1019534	335
	LAHC_HC	2974072	47	3489308	83
	LAHC_TS	2974072	109	**5222996**	94
	LAHC_SA	3346932	46	3688489	74
	GA_HC	**5373412**	4343	3473423	4245
	GA_TS	2974072	8	3606269	5472
	GA_SA	2974072	8	4619587	1265
	GA_LAHC	2974072	3	1673008	270

Table 5.47: Performance of the algorithms at FA stage for Hall 5

Hall	Algorithms	Male		Female	
		Fitness Values	Time (Minutes)	Fitness Values	Time (Minutes)
Hall 5	HC	1959575	61	2819480	40
	TS	248210	35	2780270	52
	SA	249013	39	2645386	49
	LAHC	254055	**24**	2819480	**42**
	GA	1307800	364	872984	371
	LAHC_HC	687098	22	2817099	61
	LAHC_TS	**2476189**	88	**5511655**	84
	LAHC_SA	611184	34	2748806	49
	GA_HC	1664962	5885	3092136	2567
	GA_TS	424321	35	3430390	3290
	GA_SA	386057	28	3433795	3557
	GA_LAHC	267918	16	1976166	269

Table 5.48: Performance of the algorithms at FA stage for Hall 6

Hall	Algorithms	Male		Female	
		Fitness Values	Time (Minutes)	Fitness Values	Time (Minutes)
Hall 6	HC	309552	**50**	115698	**21**
	TS	309552	61	108783	27
	SA	309552	55	103910	48
	LAHC	309552	52	107239	28
	GA	157260	381	27917	342
	LAHC_HC	309552	49	257298	94
	LAHC_TS	309552	120	**286953**	327
	LAHC_SA	396232	57	206752	57
	GA_HC	**636756**	4290	274512	1372
	GA_TS	309552	48	189508	1795
	GA_SA	309552	66	200687	1430
	GA_LAHC	309552	52	137085	221

5.5 Summary

This chapter has presented the results of the five heuristic algorithms and their hybrids applied to the HSAP. The exact solution obtained using AIMMS was only performed for the first (CA) stage being the most important stage that determines who is to be accommodated and who is to be excluded. The remaining two stages (HA and FA) deal with the distribution of students into the halls and floors respectively hence the heuristic algorithms were employed here. The principal objective of the extensive application of heuristics is to determine from among the various combinations which is the best that can be recommended for actual implementation of the HSAP. The performances of the algorithms were not the same at all stages of allocations and for all three datasets. This is an expected outcome as essentially all the heuristics employed are stochastic in nature.

From the results obtained, LAHC_TS provided the best results in 21 out of 36 cases based on the fitness values as the performance metric especially at all stages. Out of the remaining 15 cases, the performance of LAHC_TS is quite comparative to the best algorithm. Moreover, where the LAHC_TS is not the best performing algorithm, it is somewhat good enough to be considered as a possible technique to implement while developing a decision support system for the HSAP that has similar dataset characteristics and constraints.

Also, in most cases, the hybrid algorithms provide outstanding feasible solutions. In addition, at the CA stage, HC was the best performer in two out of six cases. At the HA and FA stages, the hybrid algorithms have the best solutions compared to other “pure” algorithms. This confirms

the underlying idea of synergizing the strength of the underlying heuristics into this hybrid for better performance. In addition, the results show clearly that hybrid algorithms are the best performing algorithms for this instance of the SAP given the three data sets.

The time required for the manual computation of the distributions can be enormous as it takes days and in some cases weeks to compute. However, the results show that an automated solution with the underlying heuristics will offer a promising alternative in the allocation process. This research work has thus further confirmed the viability and efficiency of applying heuristics in tackling the HSAP.

CHAPTER 6

CONCLUSION AND FURTHER WORK

6.1 Summary and Conclusion

The HSAP was recently introduced in literature with a case study from Nigeria. Presently, administrators of HILs especially in Nigeria where this study is based are beginning to appreciate the need for automated solutions that incorporate efficient approximation algorithms to tackle this problem. Such solutions will enhance the efficiency, transparency and effectiveness of the decision making process involved in distributing available limited bed spaces to meet the ever increasing demand from students. Definitely, such an automated transparent solution will increase students' trust in the allocation process being executed by the university as it will ensure fairness in distribution while also enhancing the academic performance of the students. Moreover, a fair residence allocation process will enhance the smooth running of the institution as a result of high levels of student satisfaction. This study presented a further study into the viability of heuristics especially LS techniques and hybrids for solving the multi-stage HSAP that is expressed as a form of KSP.

This dissertation presents mathematical models for the multi-staged HSAP as revealed from the case study considered. The general aim of this research work is to seek novel and innovative approaches that may be used to generate even distributions of students based on a specified set of hard and soft constraints. The problem is a form of the KSP where students in various categories are to be distributed into halls and floors that have varying capacities, each category has an

associated weight (number of students to be allocated) and profit (the cost of assigning students in that category to a hall). As is the case with the underlying KSP, this is a well-known NP-hard problem; therefore, the search for efficient heuristics that give the best feasible solutions is necessary. In view of this, the research work studied the performance of a number of single-solution based heuristics, GA and hybrids to determine which of them yields the best performance in terms of proffering solutions to the HSAP. Of note is the LAHC approach which recently attracted large scale interest amidst metaheuristics researchers seeking solutions to complex real-world COPs (Burke & Bykov, 2010).The motivation for using these techniques lie in the fact that they have proved to be successful in solving many well-known KSPs and BPPs. Some have been successfully applied to similar SAPs in previous researches as revealed in literature but none has applied these techniques to solve the HSAP hence the significance of this study.

Results obtained have shown the effectiveness of employing heuristics in determining near-optimal solutions for the HSAP. This study has also shown that hybridizing heuristics in a way that combines their strong points can help to improve their efficiency and performance as the hybrid algorithms implemented in this thesis clearly outperformed other algorithms especially at the HA and FA stages. In addition, the LAHC_TS had an exceptional performance compared to other algorithms. These results establish the viability and justification for applying heuristics

6.2 Further Works

HSAP is still a relatively new field of study in literature with much still to be done both on the case study, benchmarking, modelling, and solution techniques for the problem. Moreover,

constraints vary from one HIL to another especially across countries. For example, the requirements and criteria for allocating students in halls of residence in South Africa are slightly different from those obtainable in Nigeria where this case study is based. There is therefore the need for further future studies in this area as it applies to HILs across countries in order to be able to establish a more generalized model for the HSAP. Moreover, in this current study, the possibility of harmonising the multi-level models of the HSAP into a single model might provide more insight in proffering solutions to this problem.

Although, it is very challenging to obtain real-life archived data from past manually computed allocation distributions due to the problem of poor administrative processes in many institutions, it will be noteworthy to consider the possibility of harmonising the multi-level models of the HSAP into a single model over time in order to have worthy benchmark for subsequent results. Also, various opportunities abound to study the performance of other classes of metaheuristics and computational intelligence techniques in addressing the HSAP. No study as yet has examined the performance of swarm intelligence techniques, intelligent solutions such as artificial neural networks, self-organizing map, etc in solving HSAP. Furthermore, researchers involved in mathematical modelling can further assist in specifying a strong mathematical model needed to address the problem and to prove that the HSAP is essentially NP-hard in nature.

References

Abramson, D. (1991). Constructing School TimeTables Using Simulated Annealing: Sequential and Parallel Algorithms. *Management Science. 37*, 98-113.

Abuhamdah, A. (2010). Experimental result of late acceptance randomized descent algorithm for solving course timetabling problems. *IJCSNS International J. of Compu. Sci. and Network Security 10* , 192-200.

Adewumi, A., & Ali, M. M. (2009). *A Hierarchical Heuristic Strategy for Hostel Space Allocation Problem.* Unpublished manuscript under revision to the ToP (Operations Research) journal.

Adewumi, A., & Ali, M. M. (2010). A multi-level genetic algorithm for a multi-stage space allocation problem. *ELSEVIER 51 (2010), 109-126.*

Adewumi A. O. (2010). Some Improved Genetic-Algorithms Based Heuristics for Global Optimisation with Innovative Applications. PhD Thesis, School of Computational and Applied Mathematics, University of Witwatersrand, South Africa.

Adewumi, A. O., Sawyerr, B., & Ali, M. M. (2009). A Heuristics Approach to the University Timetabling Problems. *Engineering Computations, Emerald Publishers, UK.*

Akio, I., Etsuko, N., & Stratos, P. (2001). The dynamic berth allocation problem for a container port. *Transportation Research Part B 35 (2001) 401±417. Ó 2001 Elsevier Science Ltd.*

Alitheia Capital. (2012). Student Housing - An Emerging Real Estate Asset Class. UK: Alitheia Research Group.

Battiti, R., Brunato, M., & Mascia, F. (2008). Reactive Search and Intelligent Optimization. Springer Verlag .

Beasley, D., Bull, D. R., & Martin, R. R. (1993). An Overview of Genetic Algorithms: Part 1, Fundamentals, University Computing. 15. 58-69.

Bouleimen, K., & Lecocq, H. (2003). A New Efficient Simulated Annealing Algorithm For the Resource-constrained Project Scheduling Problem and Its Multiple Mode Version. *European Journal of Operational Research. 149,* 268- 281.

Bremermann, H. J. (1962). Optimisation Through Evolution and Re-Combination. In: Yovits,M., Sawbi,G., and Goldstein,G. (Eds.), Self-Organising Systems, Washington, Spartan.

Burke, E., & Bykov, Y. (2008). A late acceptance strategy in hill-climbing for exam timetabling problems (extended abstract). *In Proceedings of the 7th International Conf. on the Practice and Theory of Automated Timetabling (PATAT 2008). Montreal, Canada, August*.

Burke, E., & Bykov, Y. (2010). *A Late Acceptance Strategy in Hill-Climbing for Exam Timetabling Problems.* Nottingham, UK: Automated Scheduling Optimisation and Planning Group, School of Computer Science & IT,, The University of Nottingham.

Burke, E., Cowling, P., Landa-Silva, J., & McCollum, B. (2000). Three Methods to Automate the Space Allocation Process in UK Universities. Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling, PATAT 2000, Konstanz, Germany.

Burke, E. K., Cowling, p., & Silva, j. l. (2001). Hybrid population-based metaheuristic approaches For the space allocation problem,, . Proceedings of the 2001 IEEE Congress on Evolutionary Computation , (pp. 27-30). Seoul, Korea.

Burke, E., & Kendall, G. (1999). Applying Simulated Annealing and the No Fit Polygon to the Nesting Problem, Proceedings of WMC '99 : World Manufacturing Congress, Durham, UK, 27-30 September. 51-57.

Burke, E.K. & Varley, D.B. (1998). "'Automating space allocation in higher education."' In Proceedings of the 2nd Asia Pacific Conference on Simulated Evaluation and Learning, pages 6673, Camberra, Australia.

Carnevali, L., Coletti, L., & Patarnello, S. (1985). Image Processing by Simulated Annealing. *IBM Journal of Research and Development. 29,* 569-579.

Chen, S., & Luk, B. L. (1999). Adaptive Simulated Annealing for Optimization in Signal Processing Applications, Signal Processing. 79. 117-128.

Dammak, A., Elloumi, A., Kamoun, H. (2006), "Lecture and tutorial timetabling at a Tunisia University", in Burke, E.K., Rudova, H. (Eds),*Proceedings of the of the Sixth International Conference on Practice and Theory of Automated Timetabling (PATAT 2006)*, Masaryk University, Brno, Eds, pp.384-90

Davis, L. D. (1987). Genetic Algorithms and Simulated Annealing, Pitman, London.

Davis, L. D. (1991). Handbook of Genetic Algorithms. *Van Nostrand Reinhold*.

Dowsland, K. A. (1995). Simulated Annealing. In: Reeves,C.R. (Ed.), Modern Heuristic Techniques for Combinatorial Problems. *McGraw-Hill*, 21-69.

Edmond E.D., & Maggs, R.P. (1978). How useful are queue models in port investment decisions for container berths? *Journal of the Operational Research Society*, 29:741–750.

Edmund, K. B., & Yuri, B. (2012). The Late Acceptance Hill-Climbing Heuristic. *Technical Report CSM-192, Department of Computing Science and Mathematics, University of Stirling, ISSN 1460-9673*.

Falkenauer, E. (1997). *Genetic Algorithms and Grouping Problems.* Chichester, England: 0-4: John Wiley & Sons Ltd. ISBN 978-0-471-9715.

Fomeni, F. D. (2010). Metaheuristics for Space Allocation Problems: Comprehensive Survey and Review. Cape Town: African Institute for Mathematical Sciences (AIMS), Submitted in partial of a postgraduate diploma at AIMS

Forrest, S. (1993). Genetic Algorithms: Principles of Natural Selection Applied to Computation. Science. 261. 872-878.

Fraser, A. (1957). Simulation of genetic systems by automatic digital computers. I. Introduction. *Aust. J. Biol. Sci. 10*, 484–491.

Gendreau, M. (2003). *An Introduction to Tabu Search. Metaheuristic Handbook.* Boston M. A: Kluwer Academic Publishers, 37-54.

Gendreau, M., Hertz, A., & Laporte, G. (1994). A Tabu Search Heuristic for the Vehicle Routing Problem, Management Science. 40. 1276-1290.

Giallombardo, G., Moccia, L., Salani, M., & Vacca, I. (2010). Modeling and solving the Tactical Berth Allocation Problem. *Transportation Research Part B 44 (2010) 232–245*.

Glover, F. (1977). Heuristics for Integer Programming using Surrogate Constraints, Decisions Science, 8. 155-166.

Glover, F. (1989). Tabu Search - Part 1. *ORSA Journal on Computing 1 (2)*, 190–206.

Glover, F. (1990). Tabu Search - Part 2. *ORSA Journal on Computing 2 (1)*, 4–32.

Glover, F., & Laguna, M. (1995). Tabu Search. In: Reeves,C.R. (Ed.), Modern Heuristic Techniques for Combinatorial Problems, McGraw-Hill, pp. 21-69. Glover, F. and

Laguna, M., 1997. Tabu Search, Kluwer Academic Publishers.

Glover, F., & Laguna, M. (1997). *Tabu Search.* Kluwer Academic Publisher.

Goldberg, D. E. (1989). *Genetic Algorithms in Search Optimization and Machine Learning.* Addison Wesley. p. 41. ISBN 0-201-15767-5.

Henderson, D., Jacobson, S. H., & Johnson, A. W. (2003). The Theory and Practice of Simulated Annealing. In: Glover F. and Kochenberger,G. (Eds.), Handbook of Metaheuristics, Kluwer. 287-319.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Hu, T., Kahng, A. B., & Tsao, C.-W. A. (1995). Old bachelor acceptance: a new class of non-monotone threshold accepting methods. *ORSA J. on Computing 7(4)*, 417-425.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., & Schevon, C. (1989). Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning. *Operations Research. 37*, 865-892.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., & Schevon, C. (1991). Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Colouring and Number Partitioning. *Operations Research. 39*, 378-406.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing . *Science 220 (4598): doi:10.1126/science.220.4598.671. JSTOR 1690046.PMID 17813860*, 671–680.

Landa-Silva, J. D. (2003). Metaheuristic And Multiobjective Approaches For Space Allocation, November 2003. Nottingham: Thesis submitted to the University of Nottingham for the degree of Doctor in Philosophy, School of Computer Science and Information Technology, UK.

Landa-Silva, D., & Burke, E. K. (2007). Asynchronous Cooperative Local Search for the Office-Space-Allocation Problem. INFORMS Journal on Computing, Vol. 19, No. 4, Fall, issn 1091-9856 _eissn 1526-5528 _07 _1904 _0575 , 575-587.

Landa-Silva, D., & Obit, J. H. (2011). *Comparing Hybrid Constructive Heuristics for University Course Timetabling.* Porto, Portugal: Proc. of the VII ALIO–EURO – Workshop on Applied Combinatorial Optimization.

Landa-Silva, ¨Ulker, O., & Dario. (2010). Designing Difficult Office Space Allocation Problem Instances with Mathematical Programming. Automated Scheduling, Optimisation and Planning (ASAP) Research Group, School of Computer Science, University of Nottingham.

Liu, J. (1999). The Impact of Neighbourhood Size on the Process of Simulated Annealing: Computational Experiments on the Flowshop Scheduling Problem, Computers & Industrial Engineering. 37. 285-288.

Lopes, R., & Girimonte, D. (2010). The Office-Space-Allocation Problem in Strongly Hierarchized Organizations Infrastructures and Facilities Management Division,. *European Space Research and Technology Center, . Merz (Eds.): EvoCOP, LNCS 6022, Springer-Verlag Berlin Heidelberg*, 143-153.

M&G (2009). Varsities run out of housing. Mail & Guardians (South African) Newspaper article on higher learning, September 23, pg 31.

Martello, S., & Toth, P. (1975). An Upper Bound for the Zero-one Knapsack Problem and a Branch and Bound Algorithm,. European Journal of Operational Research. 1 , 169-175.

Martello, S., & Toth, P. (1990a). Knapsack Problems: Algorithms and Computer Implementations. John Wiley & Sons, ISBN: 0-471-92420-2.

Nyonyi, Y. (2010). Modelling of Hostel Space Allocation. . Cape Tpwn: African Institute for Mathematical Sciences (AIMS), Submitted in partial of a postgraduate diploma at AIMS.

Mazzola, J. B., & Schantz, R. H. (1995). Single-facility Resource Allocation Under Capacity-based Economics and Diseconomies of Scope, Management Science. 41. 669-689.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics 21 (6): 1087. doi:10.1063/1.1699114.*

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs.* 3rd. Ed., Springer.

Michalewicz, Z., & Fogel, D. B. (2000). *How To Solve It: Model Heuristics.* Springer-Verlag.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms, Massachusetts Institute of Technology.*

Murray, K., uller, T. M., & Rudov, H. (2010). *Modeling and Solution of a Complex University Course Timetabling Problem.* West Lafayette, USA: Space Management and AcademicScheduling, Purdue University 400 Centennial Mall Drive, IN 47907-2016.

Ogbu, F. A., & Smith, D. K. (1990). The Application of the Simulated Annealing Algorithm to the Solution of the n/m/Cmax Flowshop Problem. *Computers & Operations Research. 17*, 243-253.

Oghifo, B. (2012). Ugly Face of Hostel Congestion. Thisday Newspaper Article, 18 Aug 2012. Retrieved January 24, 2013 from http://www.thisdaylive.com/articles/ugly-face-of-hostel-congestion/122670/

Oliveiraa, R. M., Maurib, G. R., & Lorenaa, L. A. (2011). Clustering Search for the Berth Allocation Problem. *http://dx.doi.org/10.1016/j.eswa.2011.11.072,*.

Ozcan, E., Birben, M., Bykov, Y., & Burke, E. K. (2009). Examination timetabling using late acceptance hyper-heuristics. *In Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC 2009), Trondheim, Norway, May 2009* , 997-1004.

Pat-Mbano, E., Alaka1, I., & Okeoma, O. (2012). Examining the physio, psycho and socio-economic implications of non-residental policy on Imo State University students.. Copyright © Canadian Academy of Oriental and Occidenta Culturel, Vol. 8, No. 2 , 170-179.

Pereira, R., Cummiskey, K., & Kincaid, R. (2010). Office Space Allocation Optimization. *Proceedings of the 2010 IEEE, Systems and Information Engineering Design Symposium, University of Virginia, Charlottesville, VA, USA.*

Reeves, C. (1995). *Modern Heuristic Techniques For Combinatorial Problems.* McGraw-Hill: ISBN: 0-07-709239-2.

Rolland, E., Pirkul, H., & Glover, F. (1996). Tabu Search for Graph Partitioning. *Annals of Operations Research. 63*, 209-232.

Sastry, K., Goldberg, D., & Kendall, G. (2005). *Genetic Algorithms. In: Burke, E. and Kendall, G. (Eds.), Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, Boston, Dordrecht.* London: Kluwer Academic Publishers.

Schimmelpfeng, K., & Helber, S. (2006). *Application of a real-world university-course timetabling model solved by integer programming.* OR Spectrum (2007) 29:783–803, DOI 10.1007/s00291-006-0074-z. Published online: 7 December 2006 © Springer-Verlag

Scholl, A., Klein, R., & Jurgens, C. (1997). BISON: A Fast Hybrid Procedure for Exactly Solving the One-dimensional Bin Packing Problem. Computers & Operations Research. 24 , 627-645.

Sechen, C., Braun, D., & Sangiovanni-Vincetelli, A. (1988). Thunderbird: A Complete Standard Cell Layout Package. *IEEE Journal of Solid-State Circuits. 23*, 410-420.

Silva, G. C., Ferreira, T. G., & Costa, T. A. (2008). An Efficient Algorithm for the Dynamic Space Allocation Problem . EngOpt 2008 - International Conference on Engineering Optimization, 01 - 05 June. Rio de Janeiro, Brazil.

Skorin-Kapov, J., & Vakharia, A. (1993). Scheduling a Flow-Line Manufacturing Cell: A Tabu Search Approach. *International Journal of Production Research. 31*, 1721-1734.

Soriano, P., & Gendreau, M. (1996). Diversification Strategies in Tabu Search Algorithms for the Maximum Clique Problems . *Annals of Operations Research. 63*, 189-207.

Syam, W. P., & Al-Harkan, I. M. (2010). Comparison of Three Meta Heuristics to Optimize Hybrid Flow Shop Scheduling Problem with Parallel Machines. World Academy of Science, Engineering and Technology, Vol. 62 , 271-278.

Taillard, E. D. (1994). Parallel Taboo Search Techniques for the Job Shop Scheduling Problem. *ORSA Journal on Computing. 23*, 108-117.

Taillard, E. D., Gambardella, L. M., Gendreau, M., & Potvin, J.-Y. (2001). Adaptive memory programming: a unified view of metaheuristics. *European Journal of Operations Research 135*, 1-16.

Thompson, J. M., & Dowsland, K. A. (1998). A Robust Simulated Annealing Based Examination Timetabling System. *Computers & Operations Research. 25*, 637- 648.

Tian, P., Ma, J., & Zhang, D. (1999). Application of the Simulated Annealing Algorithm to the Combinatorial Optimisation Problem with Permutation Property: An Investigation of Generation Mechanism. *European Journal of Operational Research. 118*, 81-94.

Ulker, O. ¨., & Landa-Silva, D. (2010a). *A 0/1 Integer Programming Model for the Office Space Allocation Problem.* Electronic Notes in Discrete Mathematics 36 (2010) 575–582, © 2010 Elsevier B.V. All rights reserved, www.elsevier.com/locate/endm doi:10.

Ulker, O. ¨., & Landa-Silva, D. (2012). Evolutionary Local Search for Solving the Office Space Allocation Problem,. *IEEE World Congress on Computational Intelligence* (pp. 10-15). Brisbane, Australia: WCCI .

Ulker, O., & Landa-Silva, D. (2010b). *Designing Difficult Office Space Allocation Problem Instances with Mathematical Programming.* Nottingham, UK: Automated Scheduling, Optimisation and Planning (ASAP) Research Group, School of Computer Science, University of Nottingham,.

Verstichel, J., & Berghe, G. V. (2009). A late acceptance algorithm for the lock scheduling problem. *Logistic Management 2009 (5)*, 457-478.

White, G.M., Zhang, J. (1998), "Generating complete university timeTables by combining tabu search with constraint logic", in Burke, E., Carter, M. (Eds),*Lecture Notes in Computer Science*, Springer Verlag, Berlin and Heidelberg, Eds, Vol. 1408 pp.187-98.

Widmer, M., & Hertz, A. (1989). A New Heuristic Method for the Flow Shop Sequencing Problem. *European Journal of Operational Research. 41*, 186-193.

Wikipedia. (2012). Retrieved December 15, 2012, from Wikipedia Online Resources: http://en.wikipedia.org/wiki/Optimization

Yusuff, O. S. (2011). Students Access to Housing: A Case of Lagos State University students-Nigeria. Journal of Sustainable Development Vol. 4, No. 2; Published by Canadian Center of Science and Education 107.

Zahiri, S.-H. (2009). Fuzzy Multi-Objective PSO, an Approach for Office Space Allocation. *Iranian Journal Of Electrical And Computer Engineering, Vol. 8, No. 2, Summer-Fall*.

Appendix A

Table 1: Overview of the Hall Capacities

Zone (Area)	Hostel ID	Sex	Capacity
A	HA1	Male	660
	HA2	Male	444
	HA3	Female	866
B	HB1	Male	800
	HB2	Female	764
	HB3	Female	276
	HB4	Female	524
	HB5	Male	968
C	HC1	Male	526
	HC2	Female	512
	HC3	Female	646
	HC4	Male	512

Source: (Adewumi & Ali, 2010)

Table 2: Summary of Hall Facilities (Capacity)

	ZONE			
	A	B	C	Total
Female	866	1 564	1 158	**3 588**
Male	1 104	1 768	1 038	**3 910**
Total	**1 970**	**3 332**	**2 196**	**7 498**

Source: (Adewumi & Ali, 2010)

Table 3: Summary of Constraints/Requirements

Level	**Constraints**		**Classification**
Category Allocation	a.	All students in Fo, Ht and Sp categories must be allocated.	Hard
	b.	As many Fy, Sc, Fr, Ds, and Ot as possible should be allocated in this order of priority.	Soft
Hall Allocation	a.	Students in Ht, Sc and Sp must be allocated to designated hostels (see Table 4)	Hard
	b	Allocation for the remaining categories must be in the stated order of priority	Soft
Block/Floor Allocation	a.	Ht category should be allocated to the lowest floor possible in their assigned hall	Soft
	b.	Fy category should be allocated to the highest possible floor in a hall	Soft

Fo, Ht, Sp, Fr, Fy, etc. represents Foreign, health, Sport, Fresh, and Final student categories respectively

Source: (Adewumi & Ali, 2010)

Table 4: Specified Halls for certain Categories

CATEGORY	SPECIFIED HALLS	
	MALE	FEMALE
Ht	HA1	HA3
Sc	HA2	HA3
Sp	HC1	HC2

Source: (Adewumi & Ali, 2010)

Appendix B

Table 1 contains the number of applicant per category while the rest give the number of block/floor (with capacity) for various halls. A label **A 0** implies block A floor 0 (in this thesis it implies block 1 floor 1). **Cap** stands for the capacity. **Cap** with value 0 implies that the block/floor is reserved and not to be allocated. The total capacity for the hall is given in bracket

Data Set One

Table 1: Applicants by Category

Category	**Fy**	**Fo**	**Fr**	**Ht**	**Sp**	**Sc**	**Ds**	**Ot**
Male Applicants	1 240	20	1 332	70	400	400	100	1 800
Female Applicants	1 420	25	1 367	80	500	230	60	1 000

Male Halls:

HA1 (660)		**HA2 (440)**				**HB1(800)**			
Blk/Flr	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**
A 0	18	**A 0**	0	**C 0**	0	**F 0**	20	**R 0**	20
A 1	100	**A 1**	40	**C 1**	40	**F 1**	60	**R 1**	60
A 2	100	**A 2**	40	**C 2**	40	**F 2**	60	**R 2**	60
A 3	100	**A 3**	40	**C 3**	40	**F 3**	60	**R 3**	60
A 4	12	**B 0**	0	**D 0**	0	**G 0**	20	**S 0**	20
B 0	18	**B 1**	40	**D 1**	28	**G 1**	60	**S 1**	60
B 1	100	**B 2**	40	**D 2**	28	**G 2**	60	**S 2**	60
B 2	100	**B 3**	40	**D 3**	28	**G 3**	60	**S 3**	60
B 3	100								
B 4	12								

HB5(968)				**HC1(526)**				**HC4(512)**	
Blk/Flr	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**
E 0	20	**J 2**	60	**1 0**	40	**11 0**	40	**1 0**	80
E 1	60	**J 3**	60	**2 0**	40	**12 0**	40	**1 1**	144
E 2	60	**T 0**	12	**3 0**	40	**13 0**	46	**1 2**	144
E 3	60	**T 1**	60	**4 0**	40			**1 3**	144
H 0	12	**T 2**	60	**5 0**	40				
H 1	60	**T 3**	60	**6 0**	40				
H 2	60	**W 0**	12	**7 0**	40				
H 3	60	**W 1**	60	**8 0**	40				
J 0	12	**W 2**	60	**9 0**	40				
J 1	60	**W 3**	60	**10 0**	40				

Female Halls:

HA3 (866)				HB2(764)				HB3(276)	
Blk/Flr	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**
A 0	0	**D 2**	60	**D 0**	0	**U 3**	80	**C 0**	0
A 1	30	**D 3**	60	**D 1**	32	**V 0**	40	**C 1**	30
A 2	30	**E 0**	60	**D 2**	32	**V 1**	40	**C 2**	30
B 0	40	**E 1**	60	**D 3**	32	**V 2**	40	**L 0**	40
B 1	40	**E 2**	60	**K 0**	48			**L 1**	40
B 2	40	**E 3**	60	**K 1**	80			**L 2**	40
C 0	40	**F 0**	0	**K 2**	80			**P 0**	0
C 1	40	**F 1**	30	**K 3**	80			**P 1**	24
C 2	40	**F 2**	30	**U 0**	20			**P 2**	24
D 0	0	**G 0**	43	**U 1**	80			**S 0**	0
D 1	60	**H 0**	43	**U 2**	80			**S 1**	24
								S 2	24

HB4(524)				HC3(646)		HC2(512)	
Blk/Flr	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**
A 0	30	**M 2**	32	**A 0**	60	**1 0**	80
A 1	40	**M 3**	32	**A 1**	60	**1 1**	144
A 2	40	**N 0**	32	**A 2**	60	**1 2**	144
A 3	40	**N 1**	32	**B 0**	40	**1 3**	144
B 0	30	**N 2**	32	**B 1**	100		
B 1	40	**N 3**	32	**B 2**	100		
B 2	40			**C 0**	26		
B 3	40			**C 1**	100		
M 0	0			**C 2**	100		
M 1	32						

Appendix C

Data Set Two

Table 1: Applicants by Category

Category	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot
Male Applicants	30	105	600	1860	600	1998	150	2700
Female Applicants	38	120	750	2130	345	2051	90	1500

Male Halls

HA1 (990)		**HA2 (666)**				**HB1 (1200)**			
Blk/Flr	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**
A 0	27	**A 0**	0	**C 0**	0	**F 0**	30	**R 0**	30
A 1	150	**A 1**	60	**C 1**	60	**F 1**	90	**R 1**	90
A 2	150	**A 2**	60	**C 2**	60	**F 2**	90	**R 2**	90
A 3	150	**A 3**	60	**C 3**	60	**F 3**	90	**R 3**	90
A 4	18	**B 0**	0	**D 0**	0	**G 0**	30	**S 0**	30
B 0	27	**B 1**	60	**D 1**	42	**G 1**	90	**S 1**	90
B 1	150	**B 2**	60	**D 2**	42	**G 2**	90	**S 2**	90
B 2	150	**B 3**	60	**D 3**	42	**G 3**	90	**S 3**	90
B 3	150								
B 4	18								

HB5 (1452)				**HC1 (789)**				**HC4 (768)**	
Blk/Flr	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**		**Blk/Flr**	**Cap**
E 0	30	**J 2**	90	**1 0**	60	**11 0**	60	**1 0**	120
E 1	90	**J 3**	90	**2 0**	60	**12 0**	60	**1 1**	216
E 2	90	**T 0**	18	**3 0**	60	**13 0**	69	**1 2**	216
E 3	90	**T 1**	90	**4 0**	60			**1 3**	216
H 0	18	**T 2**	90	**5 0**	60				
H 1	90	**T 3**	90	**6 0**	60				
H 2	90	**W 0**	18	**7 0**	60				
H 3	90	**W 1**	90	**8 0**	60				
J 0	18	**W 2**	90	**9 0**	60				
J 1	90	**W 3**	90	**10 0**	60				

Female Halls

HA3 (1299)				HB2(1146)				HB3 (414)	
Blk/Flr	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**
A 0	0	**D 2**	90	**D 0**	0	**U 3**	120	**C 0**	0
A 1	45	**D 3**	90	**D 1**	48	**V 0**	60	**C 1**	45
A 2	45	**E 0**	90	**D 2**	48	**V 1**	60	**C 2**	45
B 0	60	**E 1**	90	**D 3**	48	**V 2**	60	**L 0**	60
B 1	60	**E 2**	90	**K 0**	72			**L 1**	60
B 2	60	**E 3**	90	**K 1**	120			**L 2**	60
C 0	60	**F 0**	0	**K 2**	120			**P 0**	0
C 1	60	**F 1**	45	**K 3**	120			**P 1**	36
C 2	60	**F 2**	45	**U 0**	30			**P 2**	36
D 0	0	**G 0**	65	**U 1**	120			**S 0**	0
D 1	90	**H 0**	65	**U 2**	120			**S 1**	36
								S 2	36

HB4 (786)				HC3 (969)		HC2 (768)	
Blk/Flr	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**
A 0	45	**M 2**	48	**A 0**	90	**1 0**	120
A 1	60	**M 3**	48	**A 1**	90	**1 1**	216
A 2	60	**N 0**	48	**A 2**	90	**1 2**	216
A 3	60	**N 1**	48	**B 0**	60	**1 3**	216
B 0	45	**N 2**	48	**B 1**	150		
B 1	60	**N 3**	48	**B 2**	150		
B 2	60			**C 0**	39		
B 3	60			**C 1**	150		
M 0	0			**C 2**	150		
M 1	48						

Appendix D

Data Set Three

Table 1: Applicants by Category

Category	Fo	Ht	Sp	Fy	Sc	Fr	Ds	Ot
Male Applicants	45	158	900	2790	900	2997	225	4050
Female Applicants	57	180	1125	3195	518	3077	135	2250

Male Halls

HA1 (1485)		**HA2 (999)**				**HB1 (1800)**			
Blk/Flr	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**
A 0	40.5	**A 0**	0	**C 0**	0	**F 0**	45	**R 0**	45
A 1	225	**A 1**	90	**C 1**	90	**F 1**	135	**R 1**	135
A 2	225	**A 2**	90	**C 2**	90	**F 2**	135	**R 2**	135
A 3	225	**A 3**	90	**C 3**	90	**F 3**	135	**R 3**	135
A 4	27	**B 0**	0	**D 0**	0	**G 0**	45	**S 0**	45
B 0	40.5	**B 1**	90	**D 1**	63	**G 1**	135	**S 1**	135
B 1	225	**B 2**	90	**D 2**	63	**G 2**	135	**S 2**	135
B 2	225	**B 3**	90	**D 3**	63	**G 3**	135	**S 3**	135
B 3	225								
B 4	27								

HB5 (2178)				**HC1(1184)**				**HC4 (1152)**	
Blk/Flr	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**
E 0	45	**J 2**	135	**1 0**	90	**11 0**	90	**1 0**	180
E 1	135	**J 3**	135	**2 0**	90	**12 0**	90	**1 1**	324
E 2	135	**T 0**	27	**3 0**	90	**13 0**	104	**1 2**	324
E 3	135	**T 1**	135	**4 0**	90			**1 3**	324
H 0	27	**T 2**	135	**5 0**	90				
H 1	135	**T 3**	135	**6 0**	90				
H 2	135	**W 0**	27	**7 0**	90				
H 3	135	**W 1**	135	**8 0**	90				
J 0	27	**W 2**	135	**9 0**	90				
J 1	135	**W 3**	135	**10 0**	90				

Female Halls

HA3 (1949)				HB2 (1719)				HB3 (621)	
Blk/Flr	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**
A 0	0	**D 2**	135	**D 0**	0	**U 3**	180	**C 0**	0
A 1	67	**D 3**	135	**D 1**	72	**V 0**	90	**C 1**	67
A 2	67	**E 0**	135	**D 2**	72	**V 1**	90	**C 2**	68
B 0	90	**E 1**	135	**D 3**	72	**V 2**	90	**L 0**	90
B 1	90	**E 2**	135	**K 0**	108			**L 1**	90
B 2	90	**E 3**	135	**K 1**	180			**L 2**	90
C 0	90	**F 0**	0	**K 2**	180			**P 0**	0
C 1	90	**F 1**	67	**K 3**	180			**P 1**	54
C 2	90	**F 2**	68	**U 0**	45			**P 2**	54
D 0	0	**G 0**	97	**U 1**	180			**S 0**	0
D 1	135	**H 0**	98	**U 2**	180			**S 1**	54
								S 2	54

HB4 (1179)				HC3 (1454)		HC2 (1152)	
Blk/Flr	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**	**Blk/Flr**	**Cap**
A 0	67	**M 2**	72	**A 0**	135	**1 0**	180
A 1	90	**M 3**	72	**A 1**	135	**1 1**	324
A 2	90	**N 0**	72	**A 2**	135	**1 2**	324
A 3	90	**N 1**	72	**B 0**	90	**1 3**	324
B 0	68	**N 2**	72	**B 1**	225		
B 1	90	**N 3**	72	**B 2**	225		
B 2	90			**C 0**	59		
B 3	90			**C 1**	225		
M 0	0			**C 2**	225		
M 1	72						