

KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY, KUMASI

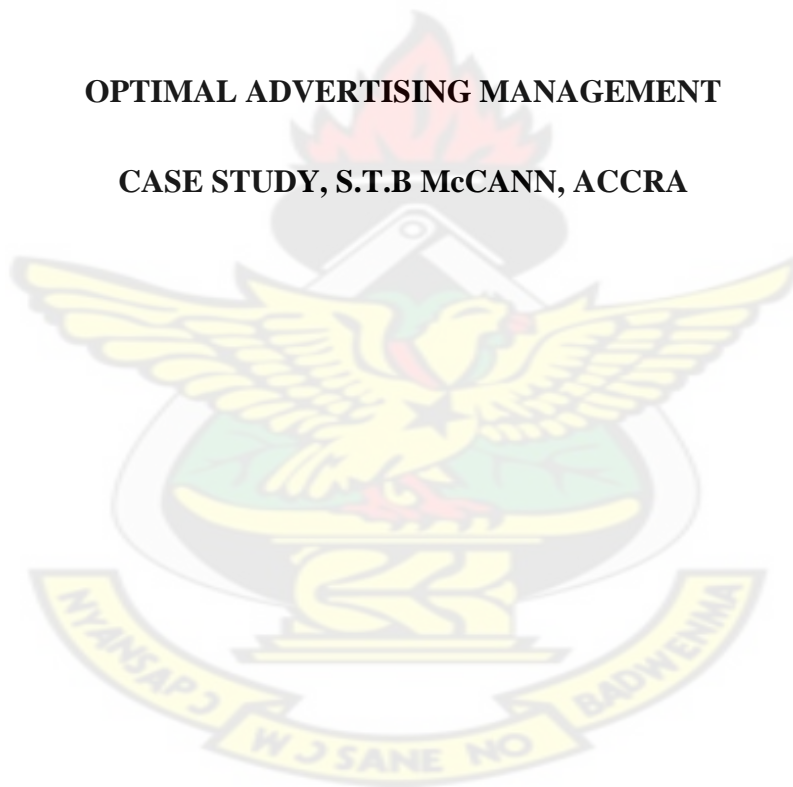
COLLEGE OF SCIENCE

DEPARTMENT OF MATHEMATICS

KNUST

OPTIMAL ADVERTISING MANAGEMENT

CASE STUDY, S.T.B McCANN, ACCRA



BY

DILYS KORKOR AGAH

MARCH, 2012

OPTIMAL ADVERTISING MANAGEMENT

CASE STUDY, S.T.B McCANN, ACCRA

A THESIS SUBMITTED TO THE GRADUATE SCHOOL BOARD

**KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY, KUMASI,
GHANA.**

**IN PARTIAL FULFILMENT OF THE AWARD OF THE DEGREE OF
MASTER OF SCIENCE (MSC) IN MATHEMATICS**

BY

DILYS KORKOR AGAH

B.Ed MATHEMATICS

MARCH, 2012

KNUST

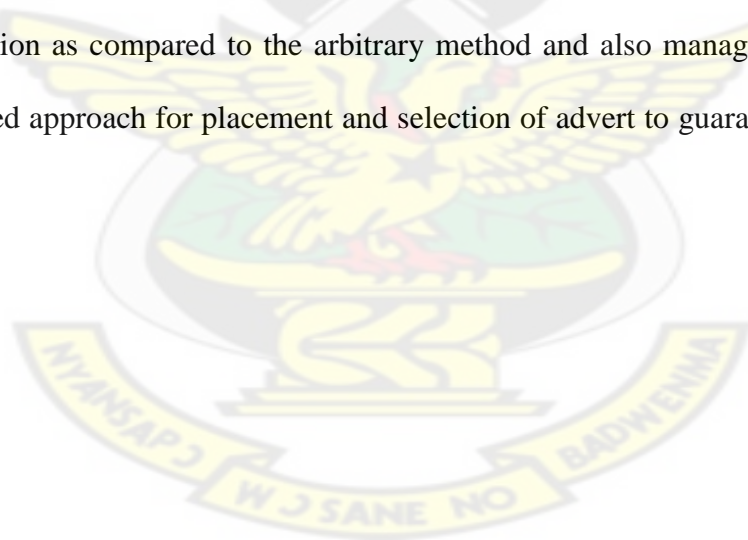
Signature

Signature

Signature

ABSTRACT

The Knapsack problem model is a general resource allocation model in which a single resource is assigned to a number of alternatives with the objective of maximizing the total return. In this study, we applied the knapsack problem model to the placement of advert slots in the media. The aim was to optimize the capital allocated for advert placements. Our study focus on the use of a simple heuristic algorithm, developed by Amponsah and Darkwah (2009) for the solution of the knapsack problem. The algorithm was coded in Fortran 90. A walk through the algorithm with our model gave the computational iterative values for the various optimal values for the various optimal solutions. The software displayed the final optimal solution for the problem. This gave an optimal reach of one hundred and fifty three thousand people at the cost of ninety seven cedis as against one hundred and fifty two thousand people at the cost of ninety eight cedis from the crude method used by the company. In the study it was observed that computer applications in computation gives a systematic and transparent solution as compared to the arbitrary method and also management will benefit from the proposed approach for placement and selection of advert to guarantee optimal reach of people.



ACKNOWLEDGEMENT

I would like to give thanks to the Almighty God for granting me the strength and knowledge for understanding this course and the completion of this write-up.

I am very grateful to my supervisor, Dr. S. K. Amponsah of the Department of Mathematics, who painstakingly read through every line of the text and offered through his rich experience, the necessary encouragement, direction, guidance and corrections for the timely completion of this thesis.

Indeed, I am indebted to entire senior members at the Department of Mathematics for their support during the period.

I would also like to express my sincere gratitude to the staff of S.T.B. McCANN especially Mr. Agbosen

I am equally indebted to my husband, siblings and wonderful daughters Michelle and Phoebe for their understanding and support during the entire course.

Finally my thanks go to all who in diverse ways helped in bringing this project to a successful end.

God Richly Bless you all.

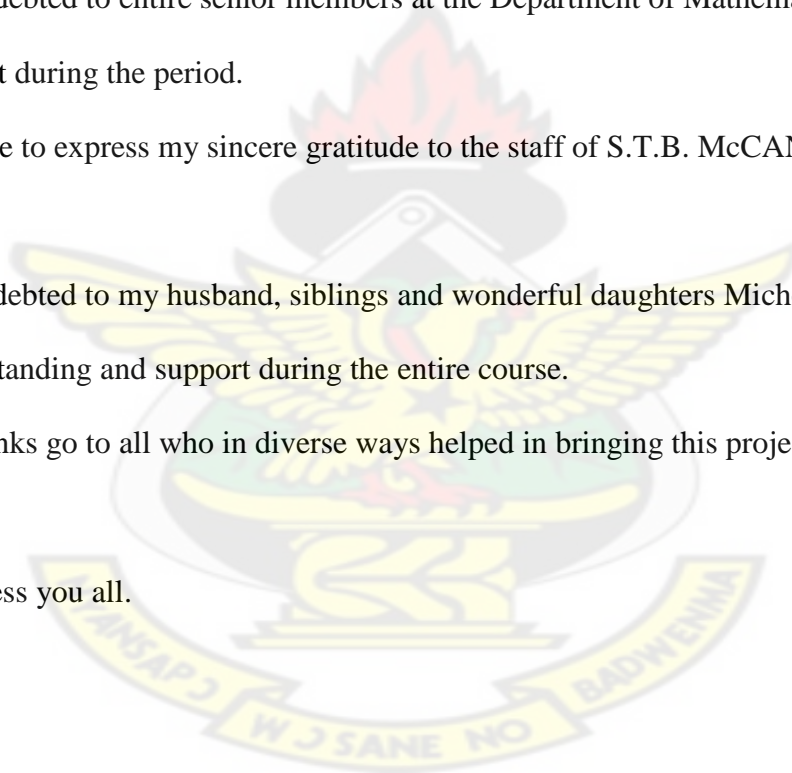


TABLE OF CONTENTS

DECLARATION	i
ABSTRACT	ii
ACKNOWLEDGEMENT	iii
TABLE OF CONTENTS	iv
LIST OF TABLES	vi
CHAPTER ONE	
1.0 INTRODUCTION	1
1.1.1 Background of Study	2
1.1.2 Problem Statement	9
1.3 Objectives	10
1.4 Methodology	10
1.5 Justification	11
1.6 Organization of the Thesis	11
1.7 Summary	11
CHAPTER TWO	
LITERATURE REVIEW	13
2.0 Introduction	13
2.1 Summary	32

CHAPTER THREE

METHODOLOGY	33
3.0 Introduction	33
3.1 Methods of solving knapsack	33
3.1.1 Branch and bound method	34
3.1.2 Dynamic Programming Method	35
3.1.3 Heuristic Scheme	35
3.1.4 Simulated Annealing	36
3.1.5 Genetic Algorithm	39
3.2 Summary	45

CHAPTER FOUR

DATA COLLECTION AND ANALYSIS	46
4.0 Introduction	46
4.1 Data Collection and Analysis	46
4.2 Results	48

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS	49
5.0 Introduction	49
5.1 Conclusions	49
5.2 Recommendations	50

REFERENCES	51
------------	----

APPENDIX_1–FORTRAN 90 CODES FOR ALGORITHM	45
---	----

APPENDIX_2 -OPTIMAL SOLUTIONS FOR THE VARIOUS ITERATIVE STAGES	56
--	----

LIST OF TABLE

4.1	Cost and Benefits of Placing an advert in a Media	56
-----	---	----

4.2	Optimal Solutions for the various iterative stages	57
-----	--	----

KNUST



CHAPTER ONE

1.0 INTRODUCTION

One of the most significant financial decisions facing individuals and institutions is the construction of optimal investment portfolios. These investment decisions are made in the present; their ultimate success or failure is only realized in the future. So this type of decision takes place under uncertainty, a common theme for financial optimization. One response to this atmosphere of uncertainty is to base judgments regarding optimality on current expectations of future returns and current perceptions of future risk. The central question of the portfolio problem is: What is the optimal way to allocate a limited amount of capital amongst a given set of investment choices? To answer this question, a process must be developed that can determine the relative weight each investment choice should have within the portfolio. That allocation should strike an acceptable balance between risk and reward. In addition, costs incurred when setting up a new portfolio or changing an existing portfolio are an important consideration. These costs must be included in any realistic decision making process (Bertsiman et al.,1999).

Allocation of resources under uncertainty is a very common problem in many real-life scenarios. Employers have to decide whether or not to hire candidates, not knowing whether future candidates will be strong or more desirable. Machines need to decide whether to accept jobs without knowledge of the importance or profitability of future jobs. Consulting companies must decide which jobs to take on, not knowing the revenue and resources associated with potential future request (Bertsiman et al.,1999).

The problem of allocating limited resources among projects is really paramount to every organization. There is then the need to allocate these resources to maximize the yield from a given investment. The aim is to select the subsets of projects which can be funded within the budget constraint. Mathematical programming models are models that can aid organizations to optimally obtain their goals through appropriate utilization and allocation of available resources (Bertsiman et al.,1999).

This thesis presents an application of mathematical programming problem, specifically knapsack problems, which provide useful information to aid decision-makers in answering the above questions. It is the work of subsequent sections and chapters to show exactly how a topic such as application of knapsack problems can be productively applied to this type of financial decision.

In this chapter of the thesis, an overview of mathematical programming model would be given; a brief description of the problem statement of the thesis is also presented together with the objectives, the methodology, the justification and the organization of the thesis.

1.1 BACKGROUND OF STUDY

Throughout human history, man has always strived to master his physical environment by making the best use of his available resources. These resources are however limited and the optimal use thereof pose potentially difficult problems. Problems of finding the best or worst situation arise constantly in daily life in a wide variety of fields that include science, engineering, economy and management. The theory of optimization attempts to find these solutions.

The theory and application of optimization is sometimes referred to as mathematical programming. Here the term programming does not refer to computer programming, but rather the allocation or composition of limited resources, although computers are extensively and in fact usually used to solve these problems. Programming problems deal with optimal allocation of limited resources such as equipment, raw materials and labor to the manufacture of one or more products. The aim is to allocate these resources in such a way that the products meet their given specifications, while at the same time maximizing some profit or minimizing cost.

To understand the concept of optimization problems, consider the case of a farmer who wants to plant a variety crops. His production is however limited by many factors. Amongst some of these are the availability of land, labour and water. After the crops have been harvested, they will be transported to different markets for sale. This give rise to the following questions: How much of each crop should he grow in order to make a maximum profit? What is the cheapest way of delivering the goods to a number of destinations? Another example is to find the shape of a string, which is anchored at

its ends along a given straight line and encloses the most area between itself and the line. This is often called Dido's problem, originating in the 18th century BC, named after a Phoenician princess Dido of Tyre, who according to the legend founded the city of Carthage in North Africa. She fled to Africa after her brother murdered her husband and appropriated her fortune. There she persuaded a local chief to sell her as much land as an ox hide could contain. She then cut the hide into thin strips and tied them together to make a cord 4km long. She laid the cord down in a semicircle with the ends touching the coast. This turns out to enclose the largest possible area and is considered as the original optimization problem(Anonymous,2011)

There are many such problems arising in a wide range of fields. These can vary in size, from problems similar to the farmer's crops involving few decision criteria to very large problems involving thousands of factors, for example, chemical reactions. Enormous efforts have been made to describe these complex human and social situations with mathematical expressions.

Initially only relatively simple problems, usually expressed using linear equations, were modeled. The evolution of mathematics and physics allowed for more complexes, but also more accurately formulated problems to be modeled using non-linear equations.

Mathematical programming is a fast growing branch of mathematics with a surprisingly short history. Most of its development has occurred during the second half of this

century. Basically, one deals with the minimization or maximization of some function subject to one or several constraints.

Mathematicians for centuries have studied linear algebraic equations. Problems came from e.g. astronomy, mechanics, geometry, and economics and so on. Many of these problems had unique solutions. However, more recently, problems have appeared from a number of applications, mathematical and non-mathematical, with typically several possible or feasible solutions to a problem. This then leads one to the question of finding a “best possible solution” among all those that are feasible in the specified sense (Geir, 1997).

Optimization is the process whereby we seek to find the best or optimal value of a function, usually subject to certain constraints or restrictions. The function with its restrictions is a mathematical optimization model that represents certain aspects of the physical environment. Optimization models are used extensively in many areas of decision making and are the cornerstone of most optimization studies. There are many methods available with which optimization problems may be solved. However, not all optimization problems can be solved efficiently with all the methods. The methods are appropriate for only certain types of problems, each designed to account for specific mathematical properties of the model. It is thus important to be able to identify the characteristics of a problem in order to find the correct solution method.

Some specific examples of optimization include :

- (i) Police patrol officer scheduling in San Francisco (linear programming), saved \$11 million per year;
- (ii) Reducing fuels costs in the electricity power industry (dynamic programming), saved over \$125 million in costs;
- (iii) Petrol blending (non-linear programming), saves \$30 million per year;
- (iv) Scheduling trucks (dynamic programming), reduced costs by \$2.5 million per year;
- (v) Maximize the expected return of several investments while at the same time minimizing risk (quadratic programming);
- (vi) Minimize the power loss in an electrical networks while satisfying the conservation of flow (quadratic programming); and
- (vii) Optimization problems can also be found in modeling of digital circuits, chemical reactions, power flow, population ecology, heartbeat and nerve impulses, crop production, food mix problems etc.

Today optimization problems arise in all sorts of areas. Modern society, with advanced technology and competitive businesses typically needs to make best possible decisions, which e.g. involve the best possible use of resources, maximizing some revenue, minimizing production or design costs, etc. In mathematical areas one may meet approximation problems like solving some equations “within some tolerance” but without using too many variables. In telecommunications, the physical design of

networks lead to many different optimization problems, e.g. that of minimizing network design costs subject to constraints reflecting that the network can support the described traffic. In economics optimization models are used for e.g. describing money transfer between sectors in society or describing the efficiency of production units.

The large amount of applications, combined with the development of fast computers, has lead to massive innovation in optimization. Today optimization may be divided into several fields, e.g. linear programming, non-linear programming, discrete optimization and stochastic optimization.

Integer programming model is a class of mathematical programming problems used to optimize linear systems that require the variables to be integers. It is the natural way of modeling many real-life and theoretical problems, including some combinatorial optimization problems and it is a broad and well-studied area.

Integer programming models are beneficial because, if one can solve them, then one is guaranteed to obtain the best solution. However, this guarantee of optimality has a computational tradeoff, and integer programming models currently may require exponential times to solve. The computational problems are so extreme that many integer programs cannot be solved, even using supercomputers (Geir, 1997).

One example of the usefulness of integer programs optimized the scheduling and deployment of San Francisco Police Department Patrol Officers (Hillier and Lieberman, 2001). The criteria used in this study were the level of public safety, level of officer's morale, and cost of operations. The computerized system that was developed used a

mathematical model to incorporate each of the goals and increased San Francisco Police Department's net income by fourteen (14) million dollars and decreased response times by 20 percent.

In addition to the above application, integer programs have been used to solve a number of real-life problems, including airline scheduling, sports scheduling (Easton et al., 2003), construction site location, manufacturing job scheduling, and telephone network optimizations (Tomastik, 1993).

A great variety of practical problems can be represented by a set of entities, each having an associated value, from which one or more subsets has to be selected in such a way that the sum of the values of the selected entities is maximized, and some predefined conditions are adhered to. The most common condition is obtained by also associating a weight to each entity and establishing that the sum of the entity sizes in each subset does not exceed some prefixed bound. These problems are generally called knapsack problems, since they recall the situation of a traveler having to fill up his knapsack by selecting from among various possible objects those which will give him the maximum comfort.

The Knapsack Problems are among the simplest integer programming problems which are NP-hard. Problems in this class are typically concerned with selecting from a set of given items, each with a specific weight and value, a subset of items whose weight sum does not exceed a prescribed capacity and whose value is maximum. The specific problem that arises depends on the number of knapsack (single or multiple) to be filled and on the number of available items of each type (bounded or unbounded). Because of their

wide range of applicability, knapsack problems have known a large number of variations such as: single and multiple constrained knapsacks, knapsack with disjunctive constraints, multidimensional knapsacks, multiple choice knapsacks, single and multiple objective knapsacks, integer, linear, non-linear knapsacks, deterministic and stochastic knapsacks, knapsacks with convex / concave objective functions, etc.

The knapsack problem is to decide what should be put in a knapsack given a weight limitation on how much can be carried. The term knapsack problem invokes the image of the backpacker who is constrained by a fixed-size knapsack and so must fill it only with the most useful or essential items. However, any problem that matches a similar analogy from any other problem area is also recognized as a knapsack problem, for example, the capital budgeting problem. The classical 0-1 Knapsack Problem arises when there is one knapsack and one item of each type. Knapsack Problems have been intensively studied over the past fifty (50) years because of their direct application to problems arising in industries and also for their contribution to the solution methods for integer programming problems. Over the last decade, knapsack problems have attracted a lot of interest for both theorists and practitioners. The theoretical interest arises mainly from their simple structure which, on the other hand allows exploitation of a number of combinatorial properties and, on the other, more complex optimization problems to be solved through a series of knapsack-type sub-problems. From the practical point of view, these problems can model many industrial situations: cutting

stock, cargo loading, and capital budgeting, to mention only the most classical applications.

1.2 PROBLEM STATEMENT

The specific form of problem that this thesis seeks to solve is to mathematically model a company's advertising problem as 0-1 knapsack problems (KP) and solve the problem.

The knapsack problem is a general resource allocation problem in which a single resource is assigned to a number of alternatives with the objective of maximizing the total return. The knapsack problem seeks to optimize a set of yes/no decisions subject to a single non-negative constraint.

Knapsack problems are widely used in financial decision making, with examples being resource allocation (Granmo et al., 2007) and portfolio management (Bertsimas et al., 1999). In resource allocation, the organization may wish to maximize its return from resources invested into each division or product subject to the total resources available. In portfolio management, the goal would be to maximize returns while minimizing risk.

1.3 OBJECTIVES

Knapsack problems have been intensively studied for the past fifty years, both because of their immediate applications in industry and financial management, and more

pronounced for theoretical reasons, as Knapsack problems frequently occur by relaxation of various integer programming problems. In such application, one need to solve a Knapsack problem each time a bounding function is derived, demanding extremely fast solution techniques.

The goal of this research is to model the advertising problem a company as a 0-1 knapsack problem and solve the problem.

1.4 METHODOLOGY

In our methodology, we shall propose the heuristic approach presented by Amponsah and Darkwah (2009) in solving our problem. First, the algorithm is presented along with relevant examples. A real life computational study is performed and a code in FORTRAN 90 programming language will be employed to evaluate this algorithm.

1.5 JUSTIFICATION

Knapsack problems are widely used in financial decision making, and very interesting from the perspective of computer science because of the time complexities in some of the well-known algorithms used in solving knapsack problems. These have made the studies of knapsack problems and their algorithms an important area of research in the contribution to academic

knowledge and the benefit of the economy as a whole, hence the reason for solving the knapsack problem.

1.6 ORGANIZATION OF THE THESIS

In Chapter 1, we presented a background study of mathematical programming model.

In Chapter 2, related work in the knapsack will be discussed.

In Chapter 3, the heuristic algorithm by Amponsah and Darkwah (2009) will be introduced and explained.

Chapter 4 will provide a computational study of the algorithm applied to our knapsack instances.

Chapter 5 will conclude this thesis with additional comments and recommendations

1.7 SUMMARY

Mathematical programming models are useful tool for modeling and optimizing real-life problems. Unfortunately, the time required to solve mathematical programming models are exponential, so real-life problems often cannot be solved. The knapsack problem is a form of mathematical programming problem that has only one constraint and can be used to strengthen cutting planes for general integer programs. In addition, knapsack problems

are widely used in financial decision making, and very interesting from the perspective of computer science since they are NP-complete. These facts make the studies of knapsack problems and their algorithms an extremely important area of research.

In the next chapter, we shall present the review of literature. This will consider the numerous studies by researchers and academicians in areas of resource scheduling and the applications of knapsack.

KNUST

CHAPTER TWO

LITERATURE REVIEW

2.0 INTRODUCTION

In this chapter, we shall discuss numerous studies by researchers and academicians in areas of resource scheduling and the applications of knapsack.

In hard real-time systems timelessness is an important as functional correctness. Such systems contain so called hard real-time tasks (HRT tasks) which must be finished by a given deadline. One of the methods of scheduling of HRT tasks is periodic loading introduced by Schweitzer et al., (1988). The authors presented an extension to that method which

allows for deterministic utilization of cache memory in hard real-time systems. It is based on a new version of the Knapsack problem named Knapsack-Lightening. In the author's presentation, the Knapsack-Lightening problem was defined, its complexity was analyzed, and an exact algorithm along with two heuristics was presented. Moreover the application of the Knapsack-lightening problem to scheduling HRT tasks was described.

Kleinberg et al., (2007) studied a model which considered the situation in which a decision-maker with a fixed budget faces a sequence of options, each with a cost and a value, and must select a subset of them online so as to maximize the total value. Such situations arise in many contexts, e.g., hiring workers, scheduling jobs, and bidding in sponsored search auctions. This problem, often called the online knapsack problem, is known to be inapproximable. Therefore, the authors made the enabling assumption that elements arrive in a random order. Hence the problem can be thought of as a weighted version of the classical secretary problem, which is called the knapsack secretary problem. Using the random-order assumption, the authors designed a constant-competitive algorithm for arbitrary weights and values, as well as a e-competitive algorithm for the special case when all weights are equal.

Knapsack problem has been widely studied in computer science for years. There exist several variants of the problem, with zero-one maximum knapsack in one dimension being the simplest one. Tauhidu (2009) studied several existing approximation algorithms for the minimization version of the problem and propose a scaling based fully polynomial time

approximation scheme for the minimum knapsack problem. The author compared the performance of this algorithm with existing algorithms, which showed that the proposed algorithm runs fast and has a good performance ratio in practice

Knapsack problems have been studied for the past few decades attracting both theorist and practitioners. The theoretical interest arises mainly from their simple structure which both allows exploitation of a number of combinatorial properties and permits more complex optimization problems to be solved through a series of knapsack type. From a practical point of view, these problems can model many industrial applications, the most classical applications being capital budget, cargo loading and cutting stock.

The collapsing knapsack problem is a generalization of the ordinary knapsack problem, where the knapsack capacity is a non-increasing function of the number of items included. Whereas previous methods on the topic have applied quite involved techniques, Ulrich et al., (1995) put forward and analyze two rather simple approaches: One approach that was based on the reduction to a standard knapsack problem, and another approach that was based on a simple dynamic programming recursion. Both algorithms have pseudo-polynomial solution times, guaranteeing reasonable solution times for moderate coefficient sizes. Computational experiments are provided to expose the efficiency of the two approaches compared to previous algorithms

The deterministic knapsack problem is a well known and well studied NP-hard combinatorial optimization problem. It consists in filling a knapsack with items out of a given set such that the weight capacity of the knapsack is respected and the total reward maximized. In the deterministic problem, all parameters (item weights, rewards, knapsack capacity) are known (deterministic). In the stochastic counterpart, some (or all) of these parameters are assumed to be random, i.e. not known at the moment the decision has to be made. Kosuch et al., (2010) studied the stochastic knapsack problem with expectation constraint. The item weights are assumed to be independently normally distributed. The authors solved the relaxed version of this problem using a stochastic gradient algorithm in order to provide upper bounds for a branch-and-bound framework. Two approaches to estimate the needed gradients are applied, one based on Integration by Parts and one using Finite Differences. Finite Differences is a robust and simple approach with efficient results despite the fact that the estimated gradients are biased; meanwhile Integration by Parts is based upon a more theoretical analysis and permits to enlarge the field of applications.

Eleni and Nicos (2010) presented a new exact tree-search procedure for solving two-dimensional knapsack problems in which a number of small rectangular pieces, each of a given size and value, are required to be cut from a large rectangular stock plate. The objective is to maximize the value of pieces cut or minimize the wastage. The authors considered the case where there are a maximum number of times that a piece may be used in a cutting pattern. The algorithm limits the size of the tree search by using a bound derived from a Lagrangean relaxation of a 0–1 integer programming formulation of the problem. Sub-gradient optimization is used to optimize this bound. Reduction tests

derived from both the original problem and the Lagrangean relaxation produce substantial computational gains. The computational performance of the algorithm indicates that it is an effective procedure capable of solving optimally practical two-dimensional cutting problems of medium size.

The bounded knapsack problem (BKP) is a generalization of the 0-1 knapsack problem where a bounded amount of each item type is available. Currently, the most efficient algorithm for BKP transforms the data instance to an equivalent 0-1 knapsack problem, which is solved efficiently through a specialized algorithm. Pisinger (1995) proposed a specialized algorithm that solves an expanding core problem through dynamic programming such that the number of enumerated item types is minimal. Compared to other algorithms for BKP, the presented algorithm uses tighter reductions and enumerates considerably less item types. Computational experiments were presented which showed that the proposed algorithm outperforms previous algorithms for BKP.

The 0-1 knapsack problem is a linear integer-programming problem with a single constraint and binary variables. The knapsack problem with an inequality constraint has been widely studied, and several efficient algorithms have been published. Balasubramanian and Sanjiv (1988) considered the equality-constraint knapsack problem, which has received relatively little attention. The authors described a branch-and-bound algorithm for this problem, and present computational experience with up to 10,000 variables. An important feature of this algorithm is a least-lower-bound discipline for candidate problem selection.

The binary knapsack problem is a combinatorial optimization problem in which a subset of a given set of elements needs to be chosen in order to maximize profit, given a budget constraint. Dash and Ghosh (2003) studied a stochastic version of the problem in which the budget is random. The authors proposed two different formulations of this problem, based on different ways of handling infeasibility, and propose an exact algorithm and a local search-based heuristic to solve the problems represented by these formulations. The authors also presented the results from some computational experiments.

The constrained compartmentalized knapsack problem is an extension of the classical integer constrained knapsack problem which can be stated as the following hypothetical situation: a climber must load his/her knapsack with a number of items. For each item a weight, a utility value and an upper bound are given. However, the items are of different classes (food, medicine, utensils, etc.) and they have to be loaded in separate compartments inside the knapsack (each compartment is itself a knapsack to be loaded by items from the same class). The compartments have flexible capacities which are lower and upper bounded. Each compartment has a fixed cost to be included inside the knapsack that depends on the class of items chosen to load it and, in addition, each new compartment introduces a fixed loss of capacity of the original knapsack. The constrained compartmentalized knapsack problem consists of determining suitable capacities of each compartment and how these compartments should be loaded, such that the total items inside all compartments does not exceed the upper bound given. The objective is to maximize the total utility value minus the cost of the compartments. This kind of problem arises in practice, such as in the cutting of steel or paper reels. Doprao and Nereu (2007) presented the problem as an integer non-linear optimization problem for which some heuristic methods are designed.

Knapsack problem has been widely studied in computer science for years. There exist several variants of the problem, with zero-one maximum knapsack in one dimension being the simplest one. Tauhidul (2009) studied several existing approximation algorithms for the minimization version of the problem and propose a scaling based fully polynomial time approximation scheme for the minimum knapsack problem. The author compared the performance of this algorithm with existing algorithms, which showed that the proposed algorithm runs fast and has a good performance ratio in practice.

Kanniappan et al., (1993) studied the problem of selecting various schemes under the integrated rural development program and to maximize the number of beneficiaries so as to optimize the annual income generated from each scheme. There are typical constraints prescribed by the government in the allocation of the funds to several schemes from the budget outlay for integrated rural development each year. Through knapsack algorithm model and data from the district rural development agency, they were able to maximize the annual income generated from the schemes.

A procurement decision model for a video rental store is modeled based on inventory management, but many classical inventory management principles are inappropriate since the commodities (movie titles) are removed from, and after a certain time period, returned to inventory. The commodities also have a decaying demand in general; hence the video rental store owner (the decision maker) is required to procure new titles periodically. There is the question of how to determine which movie titles to acquire, and

how many copies of each in order to best maximize profit. A demand function is presented, and attributes of movie titles in inventory are used to classify candidate's movie titles and predict their future demand. This allows the decision maker to select the most profitable candidate items from a list, whilst remaining within a predetermined budget. Kok et al., (2007) presented a knapsack model to achieve the above.

Scogings et al., (1995) presented a knapsack model for student enrolment at the University of Natal. The project was part of an effort to find solution to meet the demand for accommodation of lecture rooms, which is to double up on the number of time tabled period so that no lecture room will be idle for the whole day, due to the steady increase in student enrolment over the years.

Yogesh et al., (2006) addressed the problem of designing a multi-commodity network using facilities of a fixed capacity to satisfy a given set of traffic demands. This problem, called the network design problem, arises primarily in the design of high-capacity telecommunication networks. The k-partition of the network design problem graph is introduced which results in a smaller k-node network design problem. The result was that a facet inequality of the k-node problem translates into a facet of the original problem under fairly mild conditions.

Hall et al., (1992) developed a mathematical model for a project funding decision facing United State Cancer Institute. The problem was to decide which project to fund given a strict limitation on capital availability. The return for each project is calculated and the

objective is to maximize the returns from the selected projects subject to the capital constraint.

Chan et al., (2005) studied a defense modernization acquisition decision problem using multi-criteria optimization model. This model explicitly considers the diverse functions of the organization. In particular, the synergisms among the functions are modeled as a multiplicative value function. The model highlights how technology acquisitions can be affected as the priorities of each organizational function changes.

The capital budgeting problem is one of the first integer programming problems studied. It was first posed by Lorie and Savage and was called Lorie-Savage problem . Bean et al., (1987) developed a model for a multi-period version of the Lorie-Savage problem where the objective was to maximize net cash present value profit by divesting assets subject to certain lower bounds on the return on equity that companies must achieve each year.

Bhargara (1992) presented a model for the fleet mix planning of United State Coast Guard as a capital budgeting problem. This was to determine a set of new assets that can be obtained using a given capital so as to maximize the performance of the fleet.

Generating a regular season schedule is a demanding task for any sports league. In Europe, the creation of a suitable schedule for every national top soccer league not only has to address numerous conflicting inner-league requirements and preferences.

Bartsch et al., (1990) considered the case of Australia and Germany, that is the planning problem the Deutsche Fußball-Bund and the Österreichische Fußball-Bund. For both leagues they developed a binary integer programming models which yield reasonable schedules quickly.

Hospitals need to constantly produce duty rosters for its nursing staff. Appropriate and considerate scheduling of nurses can have an impact on the quality of health care, the recruitment of nurses, the development of budgets, and other nursing functions. The nurse budget problem has been the subject of many academic studies. Cheang et al., (2003) presented a binary integer programming model to solve this problem.

Bard et al., (2003) presented a full-scale integer programming model of the tour scheduling problem as it arises in the United State Postal services, and to examine several scenarios aimed at reducing the size of the workforce. The problem was formulated as a pure binary integer problem and was solved using CPLEX.

The objective of project scheduling is to determine start dates and the labour resources to be assigned to each activity in order to complete a project on time. By adjusting start dates within available slack times and altering labour levels, the daily labor demand profile can

be changed. The objective of personnel scheduling is to determine how many of each feasible workday tour are required to satisfy a given labour demand profile while minimizing labour costs and overheads. Integrating these two problems permits the simultaneous determination of start dates, labour levels, and tours for a minimum-cost and on time schedule. Bailey et al., (1995) developed single- and multiple- resource binary integer programming models for this integrated problem.

Speeding up knapsack problem, one of the NP complete problems, which could be used to design public-key cryptosystems, was presented by Lu Xin and Feng Denggu (2004) using quantum algorithm. How to use Grover's quantum searching algorithm to speed up the knapsack problem was presented based on computational complexity theory. Comparisons of quantum searching algorithm with other factoring algorithm were delivered and the factors that affected the performance of quantum algorithms were discussed from group theory point of view. The future of the quantum algorithms was also augmented in the later.

Maya and Dipti (2011) presented a research project on using Genetic Algorithms (GAs) to solve the 0-1 Knapsack Problem (KP). The Knapsack Problem is an example of a combinatorial optimization problem, which seeks to maximize the benefit of objects in a knapsack without exceeding its capacity. The author's research contains three sections: brief description of the basic idea and elements of the GAs, definition of the Knapsack Problem, and implementation of the 0-1 Knapsack Problem using GAs. The main focus of the research was on the implementation of the algorithm for solving the problem. In the program, he implemented two selection functions, roulette-wheel and group

selection. The results from both of them differed depending on whether to use elitism or not. Elitism significantly improved the performance of the roulette-wheel function. Moreover, the author tested the program with different crossover ratios and single and double crossover points but the results given were not that different.

Yunhong and Naroditskiy (2008) modeled a budget constrained keyword bidding in sponsored search auctions as a stochastic multiple-choice knapsack problem (S-MCKP) and proposed a new algorithm to solve SMCKP and the corresponding bidding optimization problem. The author's algorithm selects items online based on a threshold function which can be built/updated using historical data. Their algorithm achieved about 99% performance compared to the offline optimum when applied to a real bidding dataset. With synthetic dataset, its performance ratio against the offline optimum converges to one empirically with increasing number of periods.

An instance of the geometric knapsack problem occurs in air lift loading where a set of cargo must be chosen to pack in a given fleet of aircraft. Chocolaad (1998) presented a new heuristic to solve this problem in a reasonable amount of time with a higher quality solution than previously reported. The author also reported a new tabu search heuristic to solve geometric knapsack problems. He then employed a novel heuristics in a Master and slave relationship, where the knapsack heuristic selects a set of cargo and the packing heuristic determines if that set is feasible. The search incorporates learning mechanisms that react to cycles and thus is robust over a large set of problem sizes

Computational grids are distributed systems composed of heterogeneous computing resources which are distributed geographically and administratively. These highly scalable systems are designed to meet the large computational demands of many users from scientific and business orientations. However, there are problems related to the allocation of the computing resources which compose of a grid. Van der Vester D. C. (2008) studied the design of a Pan-Canadian grid as a binary integer programming model. The design exploits the maturing stability of grid deployment toolkits, and introduces novel services for efficiently allocating the grid resources. The changes faced by this grid deployment motivate further exploration in optimizing grid resource allocations. By applying this model to the grid allocation option, it is possible to quantify the relative merits of the various possible scheduling decisions. Using this model, the allocation problem was formulated as a knapsack problem. Formulation in this manner allows for rapid solution times and results in nearly optimal allocations.

In recent years several integrated supply chain optimization models have been studied that require the solution of knapsack problems. Examples are market selection problems where a supplier can choose which markets to serve to maximize a measure of total profit. Such problems are of independent interest but also appear as pricing problems in a branch-and-price solution approach to single-sourcing problems in which a set of retailers (or markets) needs to be assigned to a set of supply facilities to minimize total system cost. Sharkey et al., (2006) considered a large class of knapsack problems where the objective function is the sum of one linear function of the variables plus a nonlinear function of another linear function of the variables, subject to a knapsack constraint.

They showed that there always exists an optimal solution to such problems and can be generalized to the case of multiple knapsack constraints.

Last few years have seen exponential growth in the area of web applications, especially, e-commerce and web-services. One of the most important qualities of service metric for web applications is the response time for the user. Web application normally has a multi-tier architecture and a request might have to traverse through all the tiers before finishing its processing. Therefore, a request's total response time is the sum of response time at all the tiers. Since the expected response time at any tier depends upon the number of servers allocated to this tier, many different configurations (number of servers allocated to each tier) can give the same quality of service guarantee in terms of total response time. Naturally, one would like to find the configuration which minimizes the total system cost and satisfies the total response time guarantee. Zhang et al., (2004) modeled this problem as binary integer optimization problem.

Allocation of resources under uncertainty is a very common problem in many real-life scenarios. Employers have to decide whether or not to hire candidates, not knowing whether future candidates will be stronger or more desirable. Machines need to decide whether to accept jobs without knowledge of the importance or profitability of future jobs. Consulting companies must decide which jobs to take on, not knowing the revenue and resources associated with potential future requests.

More recently, online auctions have proved to be a very important resource allocation problem. Advertising auctions in particular provide the main source of monetization for a variety of internet services including search engines, blogs, and social networking sites. Additionally, they are the main source of customer acquisition for a wide array of small online business, of the networked world. In bidding for the right to appear on a web page (such as a search engine), advertisers have to trade off between large numbers of parameters, including keywords and viewer attributes. In this scenario, an advertiser may be able to estimate accurately the bid required to win a particular auction, and benefit either in direct revenue or name recognition to be gained, but may not know about the trade off for future auctions. All of these problems involve an online scenario, where an algorithm has to make decisions on whether to accept an offer, based solely on the required resource investment (or weight) and projected value of the current offer, with the total weight of all selected offer not exceeding a given budget. When the weights are uniform and equal to the weight constraint, the problems above reduces to the famous secretary problem which was first introduced by Dynkin (1963). Moshe et al., (2008), presented a binary integer programming model for this problem.

Kleinberg et al., (2007) studied the matroid secretary problem as a binary integer programming problem in which the elements of a weighted matroid arrive in a random order. As each element is observed, the algorithm makes an irrevocable decision to choose it or skip it, with the constraint that the chosen elements must constitute an independent set. The objective is to maximize the combined weight of the chosen elements.

Borgs et al., (2005) presented a model to design bidding strategies for budget-constrained advertisers in sponsored search auctions and slot selection.

The knapsack problem model has been applied to many real life applications either as a stand alone model or as a combination of models. Eilon and Williamson (1988) developed budget allocation by ranking and knapsack to solve a particular problem in determining which projects should be selected, from a given array, for implementation subject to budgetary constraint.

The strike-force asset allocation problem consists of grouping strike force assets into packages and assigning these packages to targets and defensive assets in a way that maximizes the strike force potential. Chi-Wei et al., (2001) modeled this problem as binary integer programming formulation.

The application of placement problem arises in cluster of servers that are used for hosting large, distributed applications such as internet services. Such clusters are referred to as hosting platforms. Hosting platforms imply a business relationship between the platform provider and the application providers: the latter pay the former for the resources on the platform. In return, the platform provider provides guarantees on resources availability to the applications. This implies that a platform should host only applications for which it has sufficient resources. The objective of the application placement problem is to maximize the number of applications that can be hosted on the platform while satisfying

their resource requirements. Bhuvan et al., (2005) studied the online version of the application platform problem as a binary integer programming problem.

Knapsack problems with setups find their application in many concrete industrial and financial problems. Moreover, they also arise as sub problems in a Dantzig-Wolfe decomposition approach to more complex combinatorial optimization problems, where they need to be solved repeatedly and therefore efficiently. Here, we consider the multiple-class integer knapsack problem with setups. Items are partitioned into classes whose use imply a setup cost and associated capacity consumption. Item weights are assumed to be a multiple of their class weight. The total weight of selected items and setups is bounded. The objective is to maximize the difference between the profits of selected items and the fixed costs incurred for setting-up classes. A special case is the bounded integer knapsack problem with setups where each class holds a single item and its continuous version where a fraction of an item can be selected while incurring a full setup.

Chang Sung-Ho (1998) presented k-set inequality algorithm techniques for obtaining strategies to allocate rooms to customers belonging to various market segments, considering time dependent demand forecasts and a fixed hotel capacity. This technique explicitly accounts for group and multi-night reservation requests in an efficient and effective manner. This is accomplished by combining an optimal discrete-dynamic

model for handling single-night reservation requests with a static integer programming model, developed to handle multi-night reservation requests.

The multidimensional knapsack problem (MKP) is a well-known, strongly NP-hard problem and one of the most challenging problems in the class of the knapsack problems. In the last few years, it has been a favorite playground for meta-heuristics, but very few contributions have appeared on exact methods. Renata and Grazia (2009) introduced an exact approach based on the optimal solution of sub problems limited to a subset of variables. Each sub problem is faced through a recursive variable-fixing process that continues until the number of variables decreases below a given threshold (restricted core problem). The solution space of the restricted core problem is split into subspaces, each containing solutions of a given cardinality. Each subspace is then explored with a branch-and-bound algorithm. Pruning conditions are introduced to improve the efficiency of the branch-and-bound routine.

Lin and Wei (2001) proposed an efficient linear search algorithm for solving the 0-1 knapsack problem. A net profit criterion is included in the linear search algorithm to generate a rescheduled candidate set. Four hard cases presented were tested and compared with the revised approach. Their results demonstrate that the approach proposed outperforms the previous works in terms of producing a small candidate set while retaining most of the information on optimal.

The knapsack problem is believed to be one of the “easier”-hard problems. Not only can it be solved in pseudo-polynomial time but also decades of algorithmic improvements have made it possible to solve nearly all standard instances from the literature. Pisinger (2005) gave an overview of all recent exact solution approaches and to show that the knapsack problem is still hard to solve for these algorithms for a variety of new test problems. These problems are constructed either by using standard benchmark instances with larger coefficients or by introducing new classes of instances for which most upper bounds perform badly. The first group of problems challenges the dynamic programming algorithms while the other groups of problems are focused towards branch and bound algorithms. Numerous computational experiments with all recent state-of-the-art codes are used to show that knapsack problem (KP) is still difficult to solve for a wide number of problems. One could say that the previous benchmark tests were limited to a few highly structured instances, which do not show the full characteristics of knapsack problems.

2.1 SUMMARY

In this chapter, we presented the review of literature by considering the numerous studies by researchers and academicians in areas of resource scheduling and the applications of knapsack. In the next chapter, we shall present the methods for solving the knapsack problem

CHAPTER THREE

METHODOLOGY

3.0 INTRODUCTION

This chapter provides discussions of the methods for solving knapsack problems. The knapsack problem is a specific type of integer programming problem. The knapsack formulation is the same as a basic integer program with only one constraint and binary variables. Without loss of generality, the knapsack problem is assumed to be sorted in the form $a_1 \geq a_2 \geq \dots \geq a_i$ and $\sum a_i \geq b$.

Formally, a knapsack problem is defined as:

$$\text{Maximize } c^T x$$

$$\text{Subject to: } a^T x \leq b$$

$$x \in \{0, 1\}^N.$$

Because the knapsack problem is formulated for sets of solutions containing only zeroes and ones, it is ideally suited to model decision systems. One of these types of problems is the capital budgeting problem in which a decision maker wishes to maximize profit from choosing how much to budget to a set of projects or division.

3.1 METHODS FOR SOLVING KNAPSACK PROBLEMS

There are two basic methods for solving the 0-1 knapsack problems. These are the Branch-and-Bound and the Dynamic Programming methods. However, heuristics approaches can also be employed in solving large scale knapsack problems.

3.1.1 The Branch and Bound Method

Branch and bound is a class of exact algorithms for solving optimization problems, especially integer programming problems and combinatorial optimization problems. Branch and bound uses the linear relaxation as starting point to search for the optimal integer solution. Every linear relaxation solution that is found during the branch and bound process is given a corresponding node on the branching tree. Once a node's relaxation point has been found, any variable with a fractional value may be chosen as the branching variable. Two child nodes with corresponding branches are created from this parent node. One branch requires the branching variable to be greater than or equal to its relaxation value rounded up to the nearest integer. The other branch requires the branching variable to be less than or equal to the relaxation solution rounded down to the nearest integer. Using these values, two new relaxation points are found and two more nodes are created in the tree. This process is repeated until all nodes have been fathomed.

A fathomed node is finished, and no more nodes or branches are created below any fathomed nodes. Fathoming a node in a branch and bound algorithm occurs under three circumstances. If a node is found that: (i) cannot produce a feasible solution to the linear relaxation, then that node is fathomed. (ii) returns an integer solution, then that node is fathomed. Although other feasible solutions may exist below that node, none will be

better than that node's solution. (iii) Has a linear relaxation solution with a value lower than the value of a previously discovered integer solution, then that node is fathomed.

3.1.2 Dynamic Programming Method

Dynamic programming is a method for solving optimization problems. The idea is to compute the solutions to the sub-problems once and store the solutions in a table, so that they can be reused (repeatedly) later.

The idea of developing Dynamic programming Algorithm is as follows

Step 1: Structure: Characterize the structure of an optimal solution

Decompose the problems into smaller problems, and find a relation between the structure of the optimal solution of the original problem and the solutions of the smaller problems

Step 2: Principle of Optimality: Recursively define the value of an optimal solution

Express the solution of the original problem in terms of optimal solutions for

smaller problems.

Step 3: Bottom-up computation: Compute the value of an optimal solution in a bottom-up fashion by using a table structure.

Step 4: Construction of optimal solution: Construct an optimal solution from computed information.

3.1.3 Heuristic Scheme

A heuristic scheme according to Amponsah and Darkwah (2009) may be employed to solve knapsack problems instead of branch-and-bound method. The steps for the heuristic scheme from the authors are outlined as follows:

Step 1: Input the vector of weight and item values

Step 2: Input random initial solutions S_0 and check for feasibility of S_0 by the constraint equation. If S_0 is not feasible, discard and choose another S_0

Step 3: Find a feasible solution and compute the objective function values $f(S_0)$

Step 4: Obtain a new solution S_1 by flip operation and check for feasibility, continue the flip operation until the solution S_1 obtained is feasible. Compute the objective

Function value for $f(S_1)$.

If $f(S_1) > f(S_0)$ then $S_0 = S_1$

else maintain S_0 and discard S_1 .

Step 5: Repeat Steps 3 through 4 for all feasible solutions

Step 6: Stop for non-improving solution over a number of iterations and output the best

Feasible result so far

3.1.4 Simulated Annealing

Simulated annealing is a local search algorithm capable of escaping from local optima. In its case of implementation, convergence properties and its capability of escaping from local optima has made it popular algorithm over the past years. Simulated annealing is so named because of its analogy to the process of physical annealing with solids in which a crystalline solid is heated and then allowed to cool very slowly until it achieves stable state.

At each iteration of simulated annealing, the objective function values for two solutions (the current solution and a newly generated neighboring solution) are compared. Better solutions are always accepted, while a fraction of inferior solutions are accepted in the hope of escaping local optima in search of global optima. The probability of accepting

non-improving solutions depends on a temperature parameter, which is non increasing with each iteration of the algorithm.

The key algorithm feature of simulated annealing is that it allows worse moves (i.e. moves to a solution that corresponds to a worse objective value function). As the temperature is decreased to zero, worse moves occur less frequently and the solution distribution associated with the inhomogeneous Markov chain that models the behavior of the algorithm converges to a distribution in which all the probability is concentrated on the set of globally optimal solutions implying the algorithm converges asymptotically.

To describe simulated annealing algorithm, the following definitions are needed. Let θ be the solution space: define $\vartheta(\omega)$ to be the neighborhood function for $v \in \theta$. Simulated annealing starts with an initial solution $\omega \in \theta$. A neighborhood solution $\omega^1 \in \vartheta(\omega)$ is then generated randomly in most cases. Simulated annealing is based on the metropolis acceptance criterion, which models how a thermodynamic system moves from its current solution $\omega \in \theta$ to a candidate solution $\omega^1 \in \vartheta(\omega)$ in which the energy content is being minimized. The candidate solution ω^1 is accepted as the current solution based on the acceptance probability.

In this situation, finite time implementations of simulated annealing algorithm are considered, which can no longer guarantee to find an optimal solution, but may result in faster executions without losing too much on the solution quality. Below are the steps for implementing simulated annealing:

- (i) Select an initial solution $\omega = (x_1, \dots, x_n) \in \theta$; an initial temperature $t = t$
- (ii) Control parameter value θ ; final temperature e ; a repetition schedule, M that defines the number of iterations executed at each temperature;
- (iii) Incumbent solution $\leftarrow f(\omega)$;
- (iv) Repeat;
 - (iv) Set repetition counter $m = 0$;
 - (v) Repeat;
 - (vi) Select integer i from the set $\{1, 2, \dots, n\}$ randomly;
 - (viii) If $x_i = 0$, pick up item i , i.e. set $x_i = 1$, obtain new solution ω^1 then
 - (ix) While solution ω^1 is infeasible, do
 - (x) Drop another item from ω randomly; denote the new solution as ω^1
 - (xi) Let $\Delta = f(\omega^1) - f(\omega)$
 - (xii) While $\Delta \geq 0$ or $\text{random}(0,1) < e^{(\Delta/t)}$ do $\omega \leftarrow \omega^1$
 - (xiii) Else

- (xiv) Drop item i and pick another item randomly, get new solution ω_1
- (xv) Let $\Delta = f(\omega_1) - f(\omega)$
- (xvi) While $\Delta \geq 0$ or $\text{random}(0,1) < e^{(\Delta/t)}$ do $\omega \leftarrow \omega_1$
- (xvii) End if
- (xix) If incumbent solution $< f(\omega)$, incumbent solution $\leftarrow f(\omega)$;
- (xx) $m = m + 1$;
- (xxi) Until $m = M$
- (xxii) Set $t = a * t$;
- (xxiii) Until $t < e$

A set of parameters need to be specified that govern the convergence of the algorithm, i.e.

initial temperature t_0 , temperature control α , final temperature e , and Markov chain length M , in order to study the finite time performance of simulated annealing algorithm. Here t_0 should be the maximal difference in cost between any two neighboring solutions.

3.1.5 Genetic Algorithm

A genetic algorithm (GA) can be described as an intelligent probabilistic search algorithm and is based on the evolutionary process of biological organisms in nature. In the course of evolution, natural populations evolve according to the principles of nature selection and survival of the fittest. Individuals who are most successful in adapting to their environment will have a better chance of surviving and reproducing, while individuals who are less fit will be eliminated. This means that the genes from highly fit individuals will spread to an increasing number of individuals in each successive generation. The combination of good characteristics from highly adapted parents may produce even more fit offspring. In this way, species evolve to become increasingly better adapted to the environment.

A GA simulates these processes by taking an initial population of individuals and applying genetic operators in each reproduction. In optimization terms, each individual in the population is encoded into a string or chromosome that represents a possible solution to a given problem. The fitness of an individual is evaluated with respect to a given objective function. Highly fit individuals or solutions are given opportunities to reproduce by exchanging pieces of their genetic information in a crossover procedure with other highly fit individuals. This produces new offspring solutions who share some characteristics taken from both parents. Mutation is often applied after crossover by altering some genes in the strings. The offspring can either replace the whole population (generational approach) or replace less fit individuals (steady-state approach). This evaluation-selection-reproduction cycle is repeated until a satisfactory solution is found.

Below are the steps of a simple GA

Step 1: Generate an initial population

Step 2: Evaluate fitness of individuals in the population

Step 3: Repeat

- Select individuals from the population to be parents
- Recombine (mate) parents to produce children
- Mutate the children and Evaluate fitness of the children
- Replace some or all of the population by the children until

Step 4: You decide to stop where you report the best solution so far

THE NATURE OF AN INDIVIDUAL IN OUR GA ENVIRONMENT

In the real-life we know how individuals look like. In the GA environment, how individuals look like (their representation or chromosome) is ones choice.

In the GA environment for the KP we shall choose individuals to be n bit binary strings
individual 0 1 0 0 0 1 0

Step 1: An initial population containing six individuals

Individual

1 1 1 0 0 0 0 0

2 1 0 0 1 0 0 0

3 0 0 0 0 0 0 1

4 0 0 1 0 1 0 0

5 0 1 1 0 0 0 0

6 0 1 0 0 0 1 0

has an interpretation in terms of the KP of $x_2 = x_6 = 1$ and $x_1 = x_3 = x_4 = x_5 = x_7 = 0$

Step 2: Evaluation of fitness

The objective function value ($\sum_{i=1}^n v_i x_i$) equates to how good a solution is, that is, its fitness.

In general, an individual population is randomly generated in some way.

Step 3: Selection of individuals as parents

In the real-life, individuals are independent being who for their own reasons decide to become parents. But in the GA environment we have to make a choice as to who will become a parent. In the GA environment for KP we shall choose to select parents by binary tournament selection. In binary tournament selection, we first randomly select two individuals from the population. We then select from these two the individual with the best fitness to be the first parent (individual 5 in this case).

Step 4 : Mate parents to produce children

In the real-life, parents mate to produce children.

In the GA environment for KP, we shall have a single child from two parents by uniform crossover. In uniform crossover, each bit in the child solution is created by:

repeat for each bit in turn

choose one of the two parents at random

set the child bit equal to the bit in the chosen parent

We can also have other ways as outlined below:

One-Point Crossover

In one-point crossover, we randomly select a point between two adjacent bits, cut the parents into two segments and create two children by rejoining the segments. For example, cutting parents we had before between bits 3 and 4

parent 1 0 1 1 0 0 0 0 produces segments 0 1 1 and 0 0 0 0

parent 2 0 1 0 0 0 1 0 produces segments 0 1 1 and 0 0 1 0

to give child 1 0 1 1 0 0 1 0

child 2 0 1 0 0 0 0 0

where child 1 (0110010) is composed of the first segment of parent 1 and the second segment of parent 2; child 2 (0100000) is composed of the first segment of parent 2 and the second segment of parent 1.

Restricted One-Point Crossover

From the one-point crossover example presented above we could have produced children who were identical to the parents (duplicates, clones) if we had chosen to cut the parents bits 1 and 2, bits 2 and 3: or bits 6 and 7. Restricted one-point crossover represents the cut point to ensure that the children are different from the parents. That is easily done by

simply restricting the cut point to be between the first bit where the two parents differ (bit 3 above) and the last bit where the two parents differ (bit 6 above)

(i) Fusion, as uniform crossover except that bits are taken from the parents with probabilities proportional to their fitness;

(ii) Two-point crossover, as one-point crossover (where each parent was cut into two segments) except that each parent is cut into three segments and two children produced by taken alternate segments from each parent.

Indeed, any way of combining two bit strings together could be used to produce children from two parents. Note that, one property that crossover schemes typically have in common is that bits which are the same in the parents are the same in the children.

Step 5: Mutation

Mutation corresponds to small changes that are stochastically applied to the children. Taking our child 0110010 produced by uniform crossover we could decide to make a small change, typically to randomly select one bit and to change its value (flip it). For instance we might randomly select bits 2 and flip it to give 0010010. Alternatively, we might decide (according to some probabilistic criterion) to make no mutation changes to the child. Mutation can be applied with a constant probability or with an adaptive probability that changes over the course of the algorithm (perhaps in response to the number of iterations that have passed or in response to population characteristics).

Step 6: Infeasibility

One problem that must be addressed is that (most likely) not every individual (binary bit string) represents a feasible solution in terms of the underlying problem that is being solved, for instance, for our example an individual may violate the constraints of the KP.

There are a number of strategies for dealing with constraints and infeasible solutions in GA and these are detailed below.

The first strategy is to use a representation that automatically ensures that all solutions are feasible. For some problems such representations exist, for example, the set covering problem but for the majority of the constraint problems this strategy is not possible.

The second strategy is to design a heuristic operator (often called in the literature a repair operator) that guarantees to quickly transform any infeasible solution into a feasible solution. Such a strategy is possible for KP and we illustrate this below

How to Deal with Constraints and Infeasible solutions in GA

Strategy	Description
1	To use a representation that automatically ensures all solutions are feasible
2	To design a heuristic operator that guarantees to quickly transform any infeasible solution into a feasible solution
3	To separate the evaluation of fitness and infeasibility
4	To apply a penalty function to penalize the fitness of any infeasible solutions

Heuristic Operator

For the KP, designing a heuristic operator that guarantees to quickly transform any infeasible solution into feasible solution is trivial, for example,

repeat until solution feasible:

set $x_i = 0$

KNUST

Population Replacement

We will use a steady-state population replacement strategy. With this each new child is placed in the population as soon as it is ready (after mutation and application of the heuristic operator in this case). It is common in GA to keep the population size constant hence placing the child in the population means selecting a member of the population to kill (delete). A logical approach is to kill the member of the population with the worst fitness.

3.2 SUMMARY

In this chapter, we considered the methods for solving knapsack problem. In addition the heuristic algorithm by Amponsah and Darkwah (2009) was introduced and explained.

In the next chapter, we shall present the data collection and analysis of the study.

KNUST



CHAPTER FOUR

DATA COLLECTION AND ANALYSIS

4.0 INTRODUCTION

In this chapter, we shall model a company's advertisement problem as a 0-1 knapsack problem and applied heuristics algorithm to solve our model.

The aim is to optimize the capital allocated for advertising in the business so that the business gets the best combination of adverts, through different media that would reach the largest audience possible with a minimal cost. The general practice is that most establishments do not have a well structured plan on how to allocate funds for advertising. Funds are allocated by the discretion of people or departments in charge. These methods are faulted, and are basically inefficient as funds available are not optimally utilized.

4.1 DATA ANALYSIS AND RESULTS

The Bata Shoe Company has contracted with an advertising firm (S.T.B McCANN) to determine the types and amount of advertising it should have for its stores. The three types of advertising available are radio, television commercials and newspaper ads. The retail store desires to know the number of each type of advertisement it should purchase in order to maximize exposure. It is estimated that each ad and commercial will reach the following potential audience and cost the following amount.

Table 4.1: Cost and Benefits of Placing an advert in a Media

TYPE OF ADVERTISEMENT	EXPOSURE (PEOPLE/AD COMMERCIAL) (,000)	COST GH¢(,000)
Television commercial	20	15
Radio commercial	12	8
Newspaper ad	9	4

The following resource constraints exist:

- (i) There is a budget limit of GH¢100,000 available for advertising.
- (ii) The television station has enough time available for four commercials.
- (iii) The radio station has enough time available for six radio commercials.
- (iv) The newspaper has enough space available for five ads.

The problem here is to select media types of adverts in such a way that the widest reach of people would be achieved without over shooting the amount allocated for adverts.

In comparison to the knapsack problem model, the holding capacity of the bag is the resource limit, given here as the advertising budget. The items to be considered are the different

media that can be used, the weight of any item is the cost of placing an advert using that media, and the value of the item is the reach of the media type to the people.

The problem can be modeled as:

$$\text{Maximize } R = \sum_{i=1}^n r_i m_i$$

$$\text{Subject to } \sum_{i=1}^n w_i m_i \leq W$$

$$m_i \in \{0, 1\}^N, \quad i = 1, \dots, n.$$

where;

R = Total reach

r_i = Reach of each media or item

m_i = Number of adverts placed using each media

w_i = Cost of placing an advert in each media

W = Total amount available for adverts (resource limit)

Thus,

$$\text{Maximize } R = 20 \sum_{i=1}^4 M_i + 12 \sum_{i=5}^{10} M_i + 9 \sum_{i=11}^{15} M_i$$

$$\text{Subject to } 15 \sum_{i=1}^4 M_i + 13 \sum_{i=5}^{10} M_i + 4 \sum_{i=11}^{15} M_i \leq 100$$

To carry out the computation of the proposed model, we applied the heuristics algorithm coded in Fortran 90. The feature of the software permits the input data to be fixed into the code. The Software displays the final optimal solution for the problem. However, a walk through of the algorithm with our model gave the computational iterative values for the various optimal solutions as shown in Table 4.2

The Fortran 90 code is shown in Appendix_1.

RESULTS

The various feasible combinations of media types to be selected to achieve optimal reach of people at minimum cost can be seen from Table 4.2.

The best solution among them was 153 reach of people at a cost of 97 consisting of advertising on 4 TV, 6 Radios and 5 Newspaper ads thus iteration 26 in Table 4.2.

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

5.0 INTRODUCTION

We have described the advert placement and selection problem of a company as a 0-1 knapsack programming problem. We applied Heuristics algorithm to solve the company's advert placement and selection problem. Our research focused on the use of the Knapsack problem for placement and selection of adverts given a limited available fund for a particular company in Ghana. It can however be applied to any situation that can be modeled as a 0-1 knapsack problem.

5.1 CONCLUSIONS

This thesis seeks to solve a real-life problem of a Company in Ghana using heuristics algorithm. It was observed that the solution that gave maximum achievable value was {3, 4, and 5}. This means that the company should spend a total cost of ninety seven thousand Ghana cedis (GH¢97,000) to obtain an optimal reach of one hundred and fifty three thousand (153,000) people, consisting of placing three TV, four radio, and five newspaper advertisement slots.

Currently, as at the time of this work, there is no such method for determining what media types to be used and in what quantity to be placed by the company. The media are chosen using guess work and by the discretion of the people in charge.

For the data used for our analysis, the company using their crude approach arrived at the following conclusion; placed a total of two TV, six radio, and five newspaper adverts, thus {2, 6, 5}. Total reach achieved was one hundred and fifty two thousand (152,000) people at the total cost of ninety eight thousand Ghana cedis (GH¢98,000).

5.2 RECOMMENDATIONS

The use of computer application in computation gives a systematic and transparent solution as compared with an arbitrary method. Using the more scientific Knapsack problem model for the placement and selection of the company's advert slot gives a better result. Management may benefit from the proposed approach for placement and selection of adverts to guarantee optimal reach of people. We therefore recommend that our Knapsack problem model should be adopted by the company for advert placement and media planning.

REFERENCES

1. Amponsah S. K. and Darkwah K.F (2008). Operational Research. KNUST, Institute of Distance Learning, p 43-72.
2. Bailey J., Alfares H., and Lin W. Y. (1995). Optimization and Heuristic models to integrate project task and manpower scheduling. Computers and Industrial Engineering, p 29, 473.
3. Balasubramanian R and Sanjiv S. (1988).An Algorithm for the 0-1 Equality Knapsack Problem. Journal of the Operational Research Society **39**, 1045–1049
4. Bard J. F., Bicini C., and. DeSilva A. H (2003). Staff scheduling at the United State Postal Service. Computers and Operations Research,p 30, 745.
5. Bartsch T. and Kroger S. (1990). Scheduling Australian and German professional soccer leagues. Journals of Operations Research Society of South Africa.

6. Bean J. C., Noon C. E. , and Salton G. J. (1987). Asset divestiture at Homart Development Company. *Interfaces*, 17(1): p 64-84.
7. Bertsimas D., Darnell C., and Soucy D.(1999). Portfolio construction through mixed-integer programming at Grantham, Mayo, Van Otterloo and Company. *Interfaces* 29, n1, Jan. – Feb. 1999,p 49-66.
8. Bhargava H. K. (1992). Fleet mix planning in the U. S. Coast Guard. *Issues and Challenges for DSS* in A. B. Whinstone, editor, recent developments in DSS, Springer-Varley, New York.
9. Bhuvan U., Rosenberg A., and Shenoy P. (2005). Application placement on a cluster of servers (extended abstract). Department of computer science, University of Massachusetts, Amherst, MA 01003.
10. Borgs C., Chayes J., Mahdian and Saberi (2005). Multi-unit auctions with budget-constraint bidders. <http://research.microsoft.com/enus/um/people/borgs>.
11. Chan Y., Disalva J. P., and Garrambone M. W. (2005). A goal-seeking approach to capital budgeting. *Socio-economic planning sciences*.
12. Chang Sung-Ho (1998). Tactical-Level Resource allocation procedure for the hotel industry. *Journals of Texas A & M Industrial and Systems Engineering*.

13. Chi-Wei Li Vincent (2001). Solving the strike force asset allocation problems, an application of 0-1 multi-dimensional knapsack problems with multiple choices. Journal of Texas A & M Industrial and Systems Engineering.
14. Chocolaad Christopher A.(1998). Solving Geometric Knapsack Problems using Tabu Search Heuristics
15. Dash S. and Ghosh D (2003). Binary knapsack problems with random budgets . Journal of the Operational Research Society.
16. Pisinger David (1995). A minimal algorithm for the bounded knapsack problems. Inform journal on computing, vol. 12, No. 1, pp. 75-82
17. Doprado M. F. and Nereu A. M (2007).The constrained compartmentalized knapsack problem. Journals of Computers and Operations research
18. Dynkin (1963). Random ordering hypothesis. <http://research.microsoft.com/en-us>.
19. Easton K., Nemhauser G., and Trick M. (2003). Solving the travelling tournament problem, a combined integer programming and constraint programming approach (practice and theory of automated timetabling IV). 4th International conference, PATAT 2002, Lecture notes in computer science vol. 2740, p. 100-109.
20. Eleni H.C. and Nicos C.(2010). An exact algorithm for general, orthogonal, two-dimensional knapsack problems. [European Journal of Operational Research](#)

21. Geir Dahl (1997). An introduction to convexity, polyhedral theory and combinatorial optimization. University of Oslo, Department of Informatics.

22. Granmo O. C., Oommen B. J. , Myrer S. A., and Olsen M. G. (2007). Learning automated-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation. IEE Transactions on systems, man and cybernetics, part B (cybernetics), 37 n1, 166-175.

23. Gutierrez and Talia (2007). Lifting general integer programs. Kansas State University Masters thesis.

24. Hall N., Hershey J, Dessler L, and Stotts R. (1992). A model for making project funding decisions at the national cancer institute. Journal of Operations Research.

25. Hillier F. S. and Lieberman G. J (2001). Introduction to Operations research. McGraw-Hill, New York 576-581.

<http://joc.journal.informs.org/content/early/2011/06/>

<http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identif>

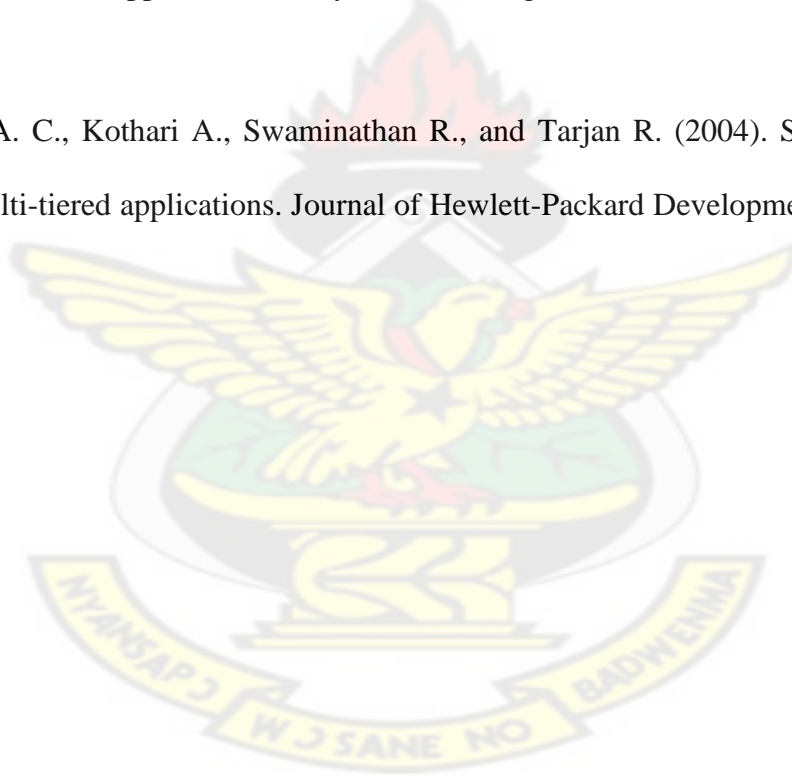
<http://research.yahoo.com/workshops/troa-2008/papers/submission>

26. Huschka Bryce (2007). Finding adjacent facet-defining inequalities. Kansas State University Masters thesis.

27. Tauhidul I. M. (2009). Approximation algorithms for minimum knapsack problem. Master's degree Thesis, university of lethbridge
28. Kanniappan P. and Thangavel K. (1993). An optimization model for selection of integrated rural development program. Journal of Operations Research Society of South Africa.
29. Kleinberg R., Babaioff M., Nicole I, and Kempe D. (2009). A knapsack secretary problem with applications. <http://research.microsoft.com/apps/pubs>.
30. Kok B. J. and J. F. Bekker (2007). A procurement decision model for a video rental store- A case study. Journal of the Operational Research Society.
31. **Lü Xin, Feng Denggu (2004).**Quantum algorithm analysis of knapsack problem JOURNAL OF BEIJING UNIVERSITY OF AERONAUTICS AND A, v30(11)
32. Maya Hristakeva and Dipti Shrestha(2011). Different Approaches to Solve the 0/1 Knapsack Problem. <www.micsymposium.org/mics
33. **Maya Hristakeva and Dipti Shrestha(2011).**Solving the 0-1 Knapsack Problem with Genetic Algorithms. <http://freetechebooks.com/file-2011/knapsack-problem>

34. Renata Minsini and Grazia Speranza (2009). An Exact Algorithm for the Multidimensional Knapsack Problem
35. Kosuch S. and Lisser A. (2010). On two-stage stochastic knapsack problems. 8th Cologne/Twente Workshop on Graphs and Combinatorial Optimization
36. Schweitzer P., Dror J. M. and Trudeau P. (1988). The periodic loading problem: Formulation and Heuristics. INFOR 26 (1), 40-62
37. Scorgings C. J. and Uys P. W (1995). Optimization of resource utilization at a university- an allocation problem. Journal of Operations Research Society of South Africa.
38. Sharkey C. T., Edwin R. H., and Geunes J. (2006). A class of nonlinear continuous knapsack problems with applications in supply chain optimization. www.ise.ufl.edu/scale
39. Tomastik R. N. (1993). The facet ascending algorithm for integer programming problems. Proceedings on the 32nd IEEE conference on decision and control, 3, 2880-2884.
40. Ulrich P. , Pisinger D., and Woeginger G.J.(1995). Simple but efficient approaches for the collapsing knapsack problem. Journal of operational Research.

41. Van derster D. C. (2008). Resource allocation and scheduling strategies using utility and knapsack problem on computational grids. Journal of Uvic Dspace.
42. Yogesh K. A. (2006). K-Partitioned based facets of the network design problem.
<http://onlinelibrary.wiley.com/doi/10.1002>.
43. Yunhong Z. and Naroditskiy V.(2008). Algorithm for Stochastic MultipleChoice Knapsack Problem and Application to Keywords Bidding
44. Zhang A. C., Kothari A., Swaminathan R., and Tarjan R. (2004). Server allocation problem for multi-tiered applications. Journal of Hewlett-Packard Development Company, L. P.



APPENDIX_1

```
#include <stdio.h>
```

```
double min (double a, double b) {
```

```
    return a < b ? a : b ;
```

```
}
```

```
struct Bounty {
```

```
    int value;
```

```
    double weight;
```

```
};
```

```
struct Bounty A = {20, 15},
```

```
    B = {12, 8}
```

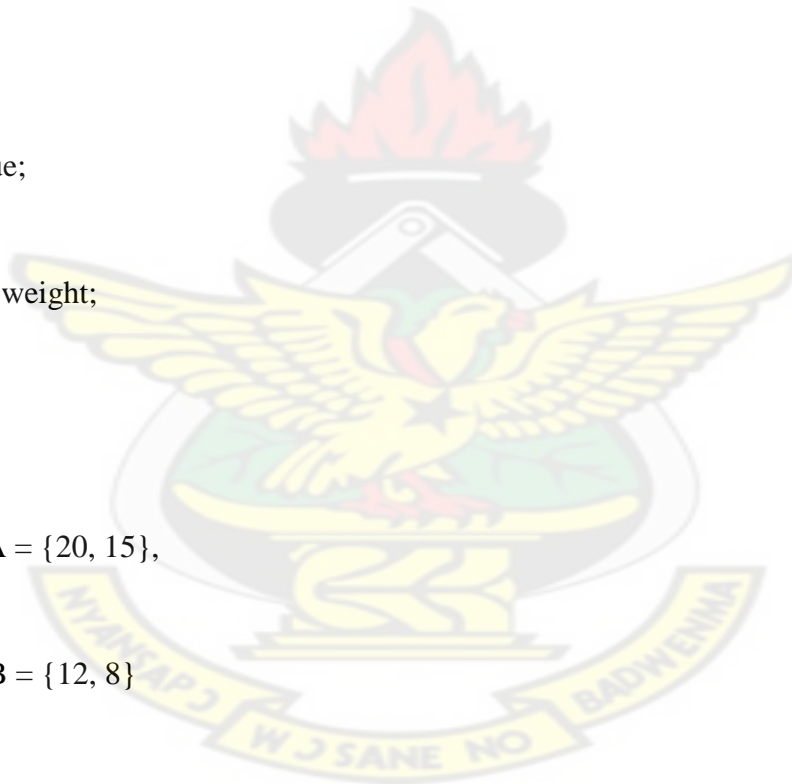
```
    C = {9, 4}
```

```
    S = {0, 100}
```

```
#define CALC(V) current.V = nA * A.V + nB * B.V + nC * C.V
```

```
int main(void) {
```

KNUST



```
int nA, nB, nC, max_A, max_B, max_C;
```

```
int best_amounts(3);
```

```
best.value = 0;
```

```
max_A = 4;
```

```
max_B = 6;
```

```
max_C = 5;
```

```
for (nA = 0; nA <= max_A; nA++) {
```

```
    for (nB = 0; nB <= max_B; nB++) {
```

```
        for (nC = 0; nC <= max_C; nC++) {
```

```
            CALC (value);
```

```
            CALC (weight);
```

```
            if (current.value > best.value && current.weight <= sack.weight) {
```

```
                best.value = current.value;
```

```
                best.weight = current.weight;
```

```
                best_amount[0] = nA;
```

```
        best_amount[1] = nB;

        best_amount[2] = nC;

    }

}

}

}

printf("Maximum value achievable is %d\n", best.value );

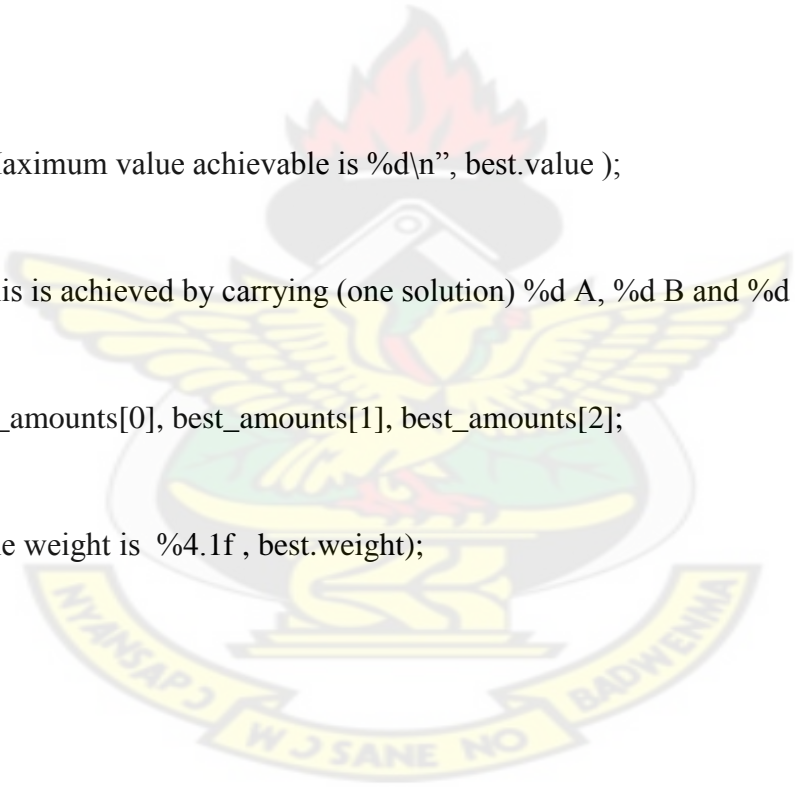
printf("This is achieved by carrying (one solution) %d A, %d B and %d C\n";

    best_amounts[0], best_amounts[1], best_amounts[2];

printf("The weight is  %4.1f , best.weight);

return 0;

}
```

The logo of KNUST (Kenya National University of Science and Technology) is visible in the background. It features a central shield with a yellow bird (likely a crane or heron) standing on a green base. Above the bird is a red flame. The shield is flanked by two yellow wings. Below the shield is a yellow banner with the text "NYANSAPU WU SANE NO BADWENNA" in black capital letters. The word "KNUST" is written in large, light gray capital letters above the logo.

APPENDIX_2

Table 4.2: Optimal Solutions for the various iterative stages

ITERATION	ADVERT PLACED	OPTIMAL REACH	OPTIMAL COST
1	{0,0,5}	45	20
2	{0,1,5}	57	28
3	{0,2,5}	69	36
4	{0,3,5}	81	44
5	{0,4,5}	93	52
6	{0,5,5}	105	60
7	{0,6,5}	117	68
8	{1,0,5}	65	35
9	{1,1,5}	72	43
10	{1,2,5}	84	51
11	{1,3,5}	96	59

12	{1,4,5}	108	67
13	{1,5, 5}	120	75
14	{1,6,5}	132	83
15	{2,0,5}	80	50
16	{2,1,5}	92	58
17	{2,2,5}	104	66
18	{2,3,5}	116	74
19	{2,4,5}	128	82
20	{2,5,5}	140	90
21	{2,6,5,}	152	98
22	{3,0,5}	105	65
23	{3,1,5}	117	73
24	{3,2,5}	129	81
25	{3,3,5}	141	89

26	{4,6,5}	153	97
27	{5,5,3}	147	97
28	{3,6,1}	141	97
29	{4,0,5}	125	80
30	{4,1,5}	137	88
31	{4,2,5}	149	96
32	{4,3,4}	152	100
33	{4,4,2}	146	100
34	{4,5,0}	140	100
35	INFEASIBLE	INFEASIBLE	INFEASIBLE
36	{5,0,5}	145	95
37	{5,1,4}	148	99
38	{5,2,2}	142	99
39	{5,3,0}	136	99

40	INFEASIBLE	INFEASIBLE	INFEASIBLE
41	INFEASIBLE	INFEASIBLE	INFEASIBLE
42	INFEASIBLE	INFEASIBLE	INFEASIBLE

KNUST

