OPTIMAL RESOURCE ALLOCATION
USING KNAPSACK PROBLEMS
A CASE STUDY OF TELEVISION ADVERTISEMENTS
AT GHANA TELEVISION (GTV)

by

Emmanuel Ofori Oppong

A Thesis submitted to the Department of Mathematics,
Kwame Nkrumah University of Science and Technology
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
(INDUSTRIAL MATHEMATICS)
Faculty of Physical Science, College of Science

February, 2009

DECLARATION

I hereby declare that this submission is my own work towards the Master of Science degree and that, to the best of my knowledge it contains no material previously published by another person nor material which has been accepted for award of any other degree of the university Except where due acknowledgement has been made in the text.

Emmanuel Ofori Oppong, PG1837007	……..	……
Student's Name & ID	Signature	Date

Certified By

Dr. S. K. Amponsah	…….	……
Supervisor's Name	Signature	Date

Certified by

Dr. S. K. Amponsah	…….	……
Head of Department's Name	Signature	Date

ACKNOWLEDGEMENT

I will like to give thanks to the Almighty God for granting me the strength and the knowledge for undertaking 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 for the completion of this thesis.

I will like also to thank Dr. F.T Oduro of the Department of Mathematics for encouraging me to embark on this course and when I went to him for advice on a postgraduate course to embark on in the Department.

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

I am equally indebted to my wife Mrs. Juliana Ofori Oppong my three children for their understanding and support for me 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.

ABSTRACT

The Knapsack Problems are among the simplest integer programs which are NP-hard. Problems in this class are typically concerned with selecting from a set of given items, each with a specified 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 knapsacks (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, knapsacks 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 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 forty (40) years because of their direct application to problems arising in industry (for example, cargo loading, cutting stock, and budgeting) and also for their contribution to the solution methods for integer programming problems. Several exact algorithms based on branch and bound, dynamic programming and heuristics have been proposed to solve the Knapsack Problems. This thesis considers the application of classical 0-1 knapsack problem with a single constraint to selection of television advertisements at critical periods such as Prime time News, new adjacencies and peak times. The Television (TV) stations have to schedule programmes interspersed with adverts or commercials which are the main sources of income of broadcasting stations. The goal in scheduling commercials is to achieve wider audience satisfaction and making maximum income from the commercials or adverts. Our approach is flexible and can incorporate the use of the knapsack for Profit maximization in the TV adverts selection problem.

Our work focuses on using a simple heuristic scheme (Simple flip) for the solution of knapsack problems. We show that the results from the heuristic method compares favourably with the well known meta-heuristic methods such as Genetic Algorithm and Simulated Annealing. The thesis shows how television advertisement at critical segments such as prime time news (19:00 GMT) and news adjacencies (five minutes before and after news time) can be prioritized to rake in the maximum returns to support operations of a national Television station (GTV). The computer

solution developed could be used for any problem that can be modeled as single 0-1 knapsack problem.

DECLARATION

I hereby declare that this submission is my own work towards the Master of Science degree and that, to the best of my knowledge it contains no material previously published by another person nor material which has been accepted for award of any other degree of the university Except where due acknowledgement has been made in the text.

Emmanuel Ofori Oppong, PG1837007	…….	……
Student's Name & ID	Signature	Date

Certified By

Dr. S. K. Amponsah	…….	……
Supervisor's Name	Signature	Date

Certified by

Dr. S. K. Amponsah	…….	……
Head of Department's Name	Signature	Date

ABSTRACT

The Knapsack Problems are among the simplest integer programs which are NP-hard. Problems in this class are typically concerned with selecting from a set of given items, each with a specified 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 knapsacks (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, knapsacks 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 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 forty (40) years because of their direct application to problems arising in industry (for example, cargo loading, cutting stock, and budgeting) and also for their contribution to the solution methods for integer programming problems. Several exact algorithms based on branch and bound, dynamic programming and heuristics have been proposed to solve the Knapsack Problems. This thesis considers the application of classical 0-1 knapsack problem with a single constraint to selection of television advertisements at critical periods such as Prime time News, news adjacencies, Break in News and peak times. The Television (TV) stations have to schedule programmes interspersed with adverts or commercials which are the main sources of income of broadcasting stations. The goal in scheduling commercials is to achieve wider audience satisfaction and making maximum income from the commercials or adverts. Our approach is flexible and can incorporate the use of the knapsack for Profit maximization in the TV adverts selection problem.

Our work focuses on using a simple heuristic scheme (Simple flip) for the solution of knapsack problems. We show that the results from the heuristic method compares favourably with the well known meta-heuristic methods such as Genetic Algorithm and Simulated Annealing. The thesis shows how television advertisement at critical segments such as prime time news (19:00 GMT) and news adjacencies (five minutes before and after news time) can be prioritized to rake in the maximum returns to support operations of a national Television

station (GTV). The computer solution developed could be used for any problem that can be modeled as single 0-1 knapsack problem.

ACKNOWLEDGEMENT

I will like to give thanks to the Almighty God for granting me the strength and the knowledge for undertaking 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 for the completion of this thesis.

I will like also to thank Dr. F.T Oduro of the Department of Mathematics for encouraging me to embark on this course and when I went to him for advice on a postgraduate course to embark on in the Department.

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

I am equally indebted to my wife Mrs. Juliana Ofori Oppong, my three children Isaac, Stephen and Rebecca 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

CHAPTER ONE

INTRODUCTION

Nearly every organization faces the problem of allocating limited resources (capital and other scarce resources including time, people) across projects or other type of investments. There is therefore the need to allocate these resources to maximize the returns from a given investment.

The goal is to select the particular subsets of projects which can be funded within the budget constraint. One of the greatest resources of broadcasting stations (both Television and Radio) is Time. The Television (TV) stations have to schedule programmes interspersed with adverts or commercials which are the main sources of income of broadcasting stations. The goal in scheduling commercials is to achieve wider audience satisfaction and making maximum income from the commercials or adverts.

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 respected. 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.

In the present survey we will adopt the following terminology. The entities will be called items and their number will be indicated by n. The value and size associated with the j^{th} item will be called profit and weight, respectively, and denoted by p_j and w_j, $(j = 1, \ldots\ldots, n)$

1.1.1 History of Broadcasting in Ghana

Broadcasting in Ghana began as a department of the Ministry of Information when it started in 1935. The ministry was responsible for the formulation of national mass communication policies and for ensuring the full and effective use of the mass media for the dissemination of information, and for economic and social development of the nation.

Radio Broadcasting was first established in Ghana in 1935 with approximately three hundred (300) subscribers in Accra. The number was low because radio sets were then rare and expensive, and was the privilege of only a rich few, mainly the expatriate community who had come from countries that already had these mass communication facilities. The brain behind the introduction of broadcasting into the country was the then Governor of the Gold Coast, Sir Arnold Hodson. Broadcasting began in Ghana essentially as a relay service, re-broadcasting programmes from the BBC World Service. A year later, the service began to expand and a re-diffusion station was opened in Cape Coast, the Central Regional capital to cater for that part of the country. Three more stations were opened the following year and a new broadcasting house built in Accra during the Second World War in 1940. It had a small 1.3KW transmitter, with which transmissions could be broadcast to neighbouring institutions. During the 1940s, broadcasting began in four of the major Ghanaian languages - Twi, Fanti, Ga and Ewe.

In 1952, the then colonial government appointed a commission to advise it on ways of improving and developing broadcasting. It was to investigate among other things the establishment and maintenance of a statutory corporation to assume direction and control of broadcasting services as was the case in parent country Britain. As a result of the Commission's report, a new broadcasting system, the national service of the Gold Coast Broadcasting System was set up in 1954.

Broadcasting became a new department distinct from the Information Services to which it had previously been attached. Broadcast content at this time was mainly governmental announcements and rebroadcasts from the BBC.

From 1956, locally produced programmes increased, educational broadcasts to schools and teacher training colleges were started and outside events were broadcast live into homes. When the Gold Coast became Ghana in 1957, the Gold Coast Broadcast System became the Ghana Broadcasting System, or as it was popularly known as Radio Ghana. Mass Communication was

embraced as a way of changing society. Broadcasting in Ghana was thus to be a public service dedicated to the enlightenment and instruction of the people. Taking into consideration that its main model was the BBC, the pioneer of public service broadcasting, it was no surprise that the public service model was adopted from the onset.

Ghana' entry into the international broadcasting scene began when in 1958 the government set up another commission to advise it on launching an external service of Radio Ghana; the External Service was inaugurated in June 1961 as a result. At the same time, television was being considered and GBC Television Service was launched on 31st July 1965. In 1997, GBC entered into an agreement with WorldSpace to provide GBC with a channel on its Afristar satellite. This capability enabled GBC to provide a 24-hour, Direct Digital Broadcasting (DDB) service over a coverage area of fourteen metre (14m) sq km, encompassing millions of radio listeners. Today, due to deregulation, availability of technology and a shift in market economy, there are five television stations in Ghana and at least seventy radio stations. Broadcasting has been privatised and commercialised bringing with it the attendant competition, issues of regulation of content and of operation.

1.1.2 TV Broadcasting Programming or Scheduling

Broadcast programming, or scheduling, is the practice of organizing television or radio programs in a daily, weekly, or season-long schedule. Modern broadcasters regularly change the scheduling of their programs to build an audience for a new show, retain that audience, or compete with other broadcasters' programs.

Television scheduling strategies are employed to give programs the best possible chance of attracting and retaining an audience. They are used to deliver programs to audiences when they are most likely to want to watch them and deliver audiences to advertisers in the composition that makes their advertising most likely to be effective (Ellis, 2000). Digitally based broadcast programming mechanisms are known as Electronic program guides.

At a micro level, scheduling is the minute planning of the transmission; what to broadcast and when, ensuring that every second of airtime is covered.

1.1.3 Television Commercial Scheduling

The main source of income for private TV stations is advertising. The broadcasting is interspersed with advertising "breaks" typically 3 minutes long. In business adverts are called "spots". Typical spots lengths are 7 seconds, 15, 22, 30, 45, 60, 90, 120. It is a rule of television advertising that competing products should not be advertised within the same break. Hence products are scheduled into clash groups and products within the same clash group should not be advertised in the same break (Brown, 1969)

1.1.4 History Of Ghana Broadcasting Corporation

- July 31 1935. Radio ZOY established. It was a small relay station installed in a bungalow near the State house in Accra.
- 1939. British Government provided funds for the building of a more fitting broadcasting House and purchased a new transmitter to carry programmes to the Country and the neighbouring West African territories.
  The Broadcasting House (BH2) was opened in 1940.
- 1946. Information Services Dept. handled administration of GBC.
- 1953. Gold Coast Broadcasting System established as a Department.
- 1956. Audience Research Department set up.
- 1956. GBC News Unit set up.
- 1958. Broadcasting House (BH-3) built.
- 1960. Dr. D. F. Coleman appointed first Ghanaian Director of Broadcasting.
- 1960. GBC joined Commonwealth Broadcasting Association.
- 1961, June 1. External Service inaugurated.
- 1962. GBC Reference Library established..
- 1965, 31 July. Ghana Television inaugurated.
- 1965. Rural Broadcasting inaugurated.
- 1967, February 1. Commercial Broadcasting introduced on additional shortwave Radio.
- 1971. Public Relations Department set up.

- 1985. Colour Television introduced.
- 1986. Accra FM inaugurated

Regional Fm Stations

- Obonu 96.5fm – Tema, Greater Accra
- Garden City Radio 92.1fm – Kumasi, Ashanti
- Radio Central 92.5fm – Cape Coast, Central
- Twin City Radio 94.7fm – Sekondi-Takoradi
- Sunrise FM 106.7fm – Koforidua, Easte
- Radio B.A.R 93.5fm – Sunyani, Brong Ahafo
- Volta Star Radio 91.1fm – Ho, Volta
- Radio Savannah 91.3fm – Tamale, Northern
- U.R.A. Radio 89.7fm – Bolgatanga, Upper East
- Radio Upper West 90.1fm – Wa, Upper West

1.1.5 Television Stations in Ghana

Currently there is one state-owned TV station, two free-to air TV channels and other five other channel using either cable /satellite broadcasting. Ghana TV (GTV) is a state-owned national TV operated by the Ghana Broadcasting Corporation. Metro TV and TV3 are free-to-air TV channel. Multichoice - provides its services through satellite, as well as Cable Gold whose service through cable serves parts of Tema, and also Fontomfom TV, which telecasts in Kumasi.V-Net TV and Fantazia TV - cable TV channel. TV AGORO (TVA) is pay-TV station which broadcast wavelengths over Accra. through bouquet of six premium channels comprising news channel CNN, music channel MCM, Cartoon Network for the kids, Turner Classic Movies (TCM) and French channel CFITV. They also broadcast two religious channels on a 24-hour basis, being Trinity Broadcasting Network (TBN) and the Catholic channel EWTN. Currently most of the channels used in Ghana are on VHF. But stations like CNN and others are broadcast on UHF channels. The trend in TV broadcast is towards use of more and more UHF channels because among other reasons, there are more available.

1.2 Broadcast Television Systems

There are several broadcast television systems in use in the world today. These are the Analogue, digital switch over and digital. In terms of ownership we have state owned television stations, private stations, pay stations.

1.2.1 Analogue Television System

An analogue television system includes several components: a set of technical parameters for the broadcast signal, a system for encoding color, and possibly a system for encoding multi-channel audio. In digital television, all of these elements are combined in a single digital transmission system. All but one analogue television system began life in monochrome. Each country, faced with local political, technical, and economic issues, adopted a color system which was effectively grafted onto an existing monochrome system, using gaps in the video spectrum (explained below) to allow the color information to fit in the channels allotted. In theory, any color system could be used with any monochrome video system, but in practice some of the original monochrome systems proved impractical to adapt to color and were abandoned when the switch to color broadcasting was made. All countries use one of three color systems: NTSC, PAL, or SECAM.

Ignoring color, all television systems work in essentially the same manner. The monochrome image seen by a camera (now, the luminance component of a color image) is divided into horizontal scan lines, some number of which make up a single image or frame. A monochrome image is theoretically continuous, and thus unlimited in horizontal resolution, but to make television practical a limit had to be placed on the bandwidth of the television signal, which puts an ultimate limit on the horizontal resolution possible. When color was introduced, this limit of necessity became fixed. All current analogue television systems are interlaced; alternate rows of the frame are transmitted in sequence, followed by the remaining rows in their sequence. Each half of the frame is called a field, and the rate at which fields are transmitted is one of the fundamental parameters of a video system. It is related to the frequency at which the electric power grid operates, to avoid flicker resulting from the beat between the television screen

deflection system and nearby mains generated magnetic fields. All digital, or "fixed pixel", displays have progressive scanning and must deinterlace an interlaced source. Use of inexpensive deinterlacing hardware is a typical difference between lower- vs. higher-priced flat panel displays (PDP, LCD, etc.). All movies and other filmed material shot at twenty-four (24) frames per second must be transferred to video frame rates in order to prevent severe motion jitter effects. Typically, for twenty-five (25) frame/s formats (countries with 50 Hz mains supply), the content is sped up, while a techniques known as "3:2 pulldown" is used for 30 frame/s formats (countries with 60 Hz mains supply) to match the film frames to the video frames without speeding up the play back. Analog television signal standards are designed to be displayed on a cathode ray tube (CRT), and so the physics of these devices necessarily controls the format of the video signal. The image on a CRT is painted by a moving beam of electrons which hits a phosphor coating on the front of the tube. This electron beam is steered by a magnetic field generated by powerful electromagnets close to the source of the electron beam.

In order to reorient this magnetic steering mechanism, a certain amount of time is required due to the inductance of the magnets; the greater the change, the greater the time it takes for the electron beam to settle in the new spot.

For this reason, it is necessary to shut off the electron beam (corresponding to a video signal of zero luminance) during the time it takes to reorient the beam from the end of one line to the beginning of the next (horizontal retrace) and from the bottom of the screen to the top (vertical retrace or vertical blanking interval). The horizontal retrace is accounted for in the time allotted to each scan line, but the vertical retrace is accounted for as phantom lines which are never displayed but which are included in the number of lines per frame defined for each video system. Since the electron beam must be turned off in any case, the result is gaps in the television signal, which can be used to transmit other information, such as test signals or color identification signals.

The temporal gaps translate into a comb-like frequency spectrum for the signal, where the teeth are spaced at line frequency and concentrate most of the energy; the space between the teeth can be used to insert a color subcarrier. Broadcasters later developed mechanisms to transmit digital information on the phantom lines, used mostly for teletext and closed captioning.

PAL-Plus uses a hidden signaling scheme to indicate if it exists, and if so what operational mode it is in. NTSC has been modified by the Advanced Television Standards Committee to support an anti-ghosting signal that is inserted on a non-visible scan line. Teletext uses hidden signaling to transmit information pages. NTSC Closed Captioning signaling uses signaling that is nearly identical to teletext signaling. All six hundred and twenty (625) line systems incorporate pulses on line twenty-three (23) that flag to the display that a 16:9 widescreen image is being broadcast, though this option is not currently used on analogue transmissions

1.2.2 The Digital Television Transition

The digital television transition (also called the digital switchover (DSO) or analog switchoff (ASO), sometimes analog sunset) is the process in which analog television broadcasting is converted to and replaced by digital television. This primarily involves both TV stations and over-the-air viewers; however it also involves content providers like TV networks, and cable TV conversion to digital cable. At the other extreme, a whole country can be converted from analogue to digital television.

In many countries, a simulcast service is operated where a broadcast is made available to viewers in both analog and digital at the same time. As digital becomes more popular, it is likely that the existing analogue services will be removed.

1.2.3 The Digital Television System

A Digital television transmission is more efficient, easily integrating other digital processes, for features completely unavailable or unimaginable with analog formats.For the end-user, digital television has potential for resolutions and sound fidelity comparable with blu-ray home video and with digital multiplexing, it is also possible to offer subchannels, distinct simulcast programming, from the same broadcaster. For government and industry, digital television reallocates the radio spectrum so that can be auctioned off by the government. In the subsequent auctions, telecommunications industries can introduce new services and products in mobile telephony, wi-fi internet, and other nationwide telecommunications projects

1.3 Problem Statement

Suppose you want to invest – all or in part- a capital of c dollars and you are considering n possible investments. Let pj be the profit you expect from investment $j, and\ wj$ the amount of dollars it requires. It is self evident that the optimal solution of the knapsack problem above will indicate the best possible choice of investment.

The objects to be considered will generally be called items and their number be indicated by n. The value and size associated with the j^{th} item will be called profit and weight, respectively, and denoted by $p_j\ and\ \ w_j,\ \ (\mathrm{j} = 1, \ldots \ldots, \mathrm{n})$.

At this point you may be stimulated to solve the problem. A naïve approach would be to program a computer to examine all possible binary vectors x, selecting the best of those which satisfy the constraint. Unfortunately, the number of such vectors is 2^n, so even a hypothetical computer, capable of one billion vectors per second, would require more than 30 years for $n = 60$, more than sixty (60) years for $n = 60$, ten centuries for n=65 and so on (Pisinger, 1995).

However, specialized algorithms can, in most cases, solve a problem with $n = 100,000$ in a few seconds on a mini/micro computer.

Again, suppose the producer of a TV programme want to select among numerous adverts for the prime time (news at 19:00 hours GMT) which is interspersed with five or six spots of adverts of not more than three minutes each.

The problem considered so far is representative of a variety of knapsack-type problems in which a set of entities are given, each having an associated value and size, and it is desired to select one or more disjoint subset so that the sum of the sizes in each subset does not exceed (or equals) a given bound and the sum of the selected values is maximized.

Knapsack problems have been intensively studied, especially in the last decade, attracting both theorists and practicians. 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: capital budgeting, cargo loading, cutting stock, to mention the most classical applications.

1.4 Objective

The main objective of this survey is to determine an effective way of scheduling commercials or adverts in the television station to achieve the maximum returns. The main source of income from the private operators of broadcasting adverts hence the need to find scientific means of selecting subsets for the numerous advert to achieve substantial income for their operations within the limited space of time. The state-owned TV station (GTV) is also required to generate additional revenue to supplement the subvention from government. The thesis is to provide computer solutions to these problems.

1.5 Justification

TV station provides both visual and audio output in the form of information, education and entertainment to the public using the scarce resources of funds available while grapping with the numerous adverts from the corporate organizations and the teaming audience. In order to justify their existence, the TV stations have to generate enough revenue from sponsored programmes and adverts to support their operation, be it private or state-owned. Without any adequate scientific method of selecting from the numerous adverts received daily the maximum returns from these may no be achieved. Each advert is charged according to the number of times to be telecast and the duration in minutes or seconds. A number of practical problem s can be formulated as problems, for example the simple capital budgeting problem of choosing which project constraint on total cost. Knapsack problems can model many other managerial and industrial situations such as cargo loading and cutting stock problems, (Salkim and Derkluyer Knapsack problems and survey).

Other applications of knapsack Problems are

- Routing of vehicles (planes, trucks etc.)
- Routing of postal workers
- Drilling holes on printed circuit board
- Routing robots through a warehouse

1.6 Methodology

The methodology employed included review of relevant literature of the types of knapsack problems and the methods employed in the solution of the knapsack problems and to develop computer solutions for faster computation of the knapsack problems of data from Ghana Television (GTV).

1.7 Scope and limitation

The problems to be considered in this survey are single 0-1knapsack problems, where one container must be filled with an optimal subset of items. The capacity of such a container will be denoted by c. The more general case where *m* containers, of capacities $\mathbf{c_i}$ (i = 1,. . . , m) , are available is referred to as multiple knapsack problems.

The computer solution developed in VB.Net programming language for the single 0-1 knapsack problems could be modified to solve multi-dimensional knapsack problems.

1.8 Organization of Thesis

Chapter 1 provides the background of the Knapsack Problems, the television industry and the methodology and justification for the use of knapsack problems to solve the TV adverts selection problem.

Chapter 2 gives a review of relevant literature on Knapsack problems applications and the solution methods that have been proposed in literature.

Chapter 3 outlines some algorithms for the solution methods such as the branch and bound, the dynamic programming, heuristic scheme, simulated annealing and Genetic algorithm.

Chapter 4 deals with the data collection and results of analysis of actual data from Ghana Television (GTV).

The final chapter draws the curtain on the conclusion and future work.

1.9 Summary

In this chapter, we discussed the formulation of the knapsack problem to the solution of the Television Advert problem, the background of television systems, the background of the case study area (Ghana Television), the methodology and the justification of the thesis.In the next chapter, we shall put forward the literature pertinent in the area of 0-1 knapsack problems.

CHAPTER TWO

LITERATURE REVIEW

Knapsack problems have been studied intensively in the past decade attracting both theorist and practitioners. The theoretical interest arises mainly from their simple structure which both allows exploitation of a number of combinational 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 budgets, cargo loading and cutting stock. In this chapter we present a review of literature on knapsack problems and applications.

Benisch et al. (2005) examined the problem of choosing discriminatory prices for customers with probabilistic valuations and a seller with indistinguishable copies of a good. They showed that under certain assumptions this problem can be reduced to the continuous knapsack problem (CKP). They presented a new fast epsilon-optimal algorithm for solving CKP instances with asymmetric concave reward functions. They also showed that their algorithm can be extended beyond the CKP setting to handle pricing problems with overlapping goods (e.g. goods with common components or common resource requirements), rather than indistinguishable goods. They provided a framework for learning distributions over customer valuations from historical data that are accurate and compatible with their CKP algorithm, and validated their techniques with experiments on pricing instances derived from the Trading Agent Competition in Supply Chain Management (TAC SCM). Their results confirmed that their algorithm converges to an epsilon-optimal solution more quickly in practice than an adaptation of a previously proposed greedy heuristic.

Pendharkar et al. (2005) described an information technology capital budgeting (ITCB) problem, and showed that the ITCB problem can be modeled as a 0–1 knapsack optimization problem, and proposed two different simulated annealing (SA) heuristic solution procedures to solve the ITCB problem. Using several simulations, they empirically compared the performance of two SA heuristic procedures with the performance of two well-known ranking methods for capital

budgeting. Their results indicated that the information technology (IT) investments selected using the SA heuristics have higher after-tax profits than the IT investments selected using the two ranking methods.

Yield management is an important issue for television advertising. Anyway, the major part of the research in revenue management focus on the airline or hotel industry. The TV advertising case has some specificities, where the most important is the decomposition of the offer into a lot of small TV breaks (about 10 spots only). Martin (2004) proposed generic solutions based on simulations and approximate dynamical programming.

Transportation programming, a process of selecting projects for funding given budget and other constraints, is becoming more complex. Zhong and Young (2009) described the use of an integer programming tool, Multiple Choice Knapsack Problem (MCKP), to provide optimal solutions to transportation programming problems in cases where alternative versions of projects are under consideration. Optimization methods for use in the transportation programming process were compared and then the process of building and solving the optimization problems discussed. The concepts about the use of MCKP were presented and a real-world transportation programming example at various budget levels were provided. They illustrated how the use of MCKP addresses the modern complexities and provides timely solutions in transportation programming practice.

The knapsack container loading problem is the problem of loading a subset of rectangular boxes into a rectangular container of fixed dimensions such that the volume of the packed boxes is maximized. A new heuristic based on the wall-building approach was proposed earlier. That heuristic divides the problem into a number of layers and the packing of layers is done using a randomized heuristic. Juraitis et al. (2006) focused on ways to find proportions of the mixture of heuristics which would lead to better performance of the algorithm. New results were compared with earlier research and some other constructive heuristics.

The performance of the corresponding algorithms was experimentally compared for homogeneous and heterogeneous instances. Proposed improvements allow achieving better filling ratio without increasing the computational complexity of the algorithm

Glickman and Allison, (1973) considered the problem of choosing among the technologies available for irrigation by tubewells to obtain an investment plan which maximizes the net agricultural benefits from a proposed project in a developing country. Cost and benefit relationships were derived and incorporated into a mathematical model which is solved using a modification of the dynamic programming procedure for solving the knapsack problem. The optimal schedule was seen to favor small capacity wells, drilled by indigenous methods, with supplementary water distribution systems.

Ferreira, (1995) presented parallel algorithms for solving a knapsack problem of size n on PRAM and distributed memory machines. The algorithms were work-efficient in the sense that they achieved optimal speedup with regard to the best known solution to this problem. Moreover, they match the best current time/memory/processors tradeoffs, while requiring less memory and/or processors. Since the PRAM is considered mainly as a theoretical model, and we want to produce practical algorithms for the knapsack problem, its solution in distributed memory machines is also studied. For the first time in literature, work-efficient parallel algorithms on local memory — message passing architectures — are given. Time bounds for solving the problem on linear arrays, meshes, and hypercubes are proved.

Bazgan et al. (2007) presented an approach, based on dynamic programming, for solving the 0/1 multi-objective knapsack problem. The main idea of the approach relies on the use of several complementary dominance relations to discard partial solutions that cannot lead to new non-dominated criterion vectors. This way, they obtained an efficient method that outperforms the existing methods both in terms of CPU time and size of solved instances.
Extensive numerical experiments on various types of instances were reported. A comparison with other exact methods was also performed.

The data association problem consists of associating pieces of information emanating from different sources in order to obtain a better description of the situation under study. This problem arises, in particular, when, considering several sensors, we aim at associating the measures corresponding to a same target. This problem, widely studied in the literature, is often stated as a

multidimensional assignment problem where a state criterion is optimized. While this approach seems satisfactory in simple situations where the risk of confusing targets is relatively low, it is much more difficult to get a correct description in denser situations. Hugot et al. (2006) proposed, to address this problem in a multiple criteria framework using a second complementary criterion, based on the identification of the targets. Due to the specificities of the problem, simple and efficient approaches can be used to generate non-dominated solutions. Moreover, they showed that the accuracy of the proposed solutions is greatly increased when considering a second criterion. A bi-criteria interactive procedure is also introduced to assist an operator in solving conflicting situations.

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 (1996) proposed a specialized algorithm that solves an expanding core problem through dynamic programming such that the number of enumerated item types is minimal. Sorting and reduction is done by need, resulting in very little effort for the preprocessing. Compared to other algorithms for BKP, the presented algorithm uses tighter reductions and enumerates considerably less item types. Computational experiments are presented, showing that the presented algorithm outperforms all previously published algorithms for BKP.

Several types of large-sized 0-1 Knapsack Problems (KP) may be easily solved, but in such cases most of the computational effort is used for sorting and reduction. In order to avoid this problem it has been proposed to solve the so-called core of the problem: a Knapsack Problem defined on a small subset of the variables. The exact core cannot, however, be identified before KP is solved to optimality, thus, previous algorithms had to rely on approximate core sizes. Pisinger (1997) presented an algorithm for KP where the enumerated core size is minimal, and the computational effort for sorting and reduction also is limited according to a hierarchy. The algorithm is based on a dynamic programming approach, where the core size is extended by need, and the sorting and reduction is performed in a similar "lazy" way. Computational experiments were presented for several commonly occurring types of data instances. Experience from these tests indicated that

the presented approach outperforms any known algorithm for KP, having very stable solution times.

Ant colony optimization algorithm is a novel simulated evolutionary algorithm, which provides a new method for complicated combinatorial optimization problems. Shuang et al. (2006) used the algorithm for solving the knapsack problem. It was improved in selection strategy and information modification, so that it can not easily run into the local optimum and can converge at the global optimum. The experiments showed the robustness and the potential power of this kind of meta-heuristic algorithm

Martello and Toth (1998) presented a new algorithm for the optimal solution of the 0-1 Knapsack problem, which is particularly effective for large-size problems. The algorithm is based on determination of an appropriate small subset of items and the solution of the corresponding "core problem": from this they derived a heuristic solution for the original problem which, with high probability, can be proved to be optimal. The algorithm incorporated a new method of computation of upper bounds and efficient implementations of reduction procedures.

The multidimensional 0–1 knapsack problem is one of the most well-known integer programming problems and has received wide attention from the operational research community during the last four decades. Although recent advances have made possible the solution of medium size instances, solving this NP-hard problem remains a very interesting challenge, especially when the number of constraints increases. Fréville surveyed the main results published in the literature and focused on the theoretical properties as well as approximate or exact solutions of this special 0–1 program.

The multidimensional 0–1 knapsack problem, defined as a knapsack with multiple resource constraints, is well known to be much more difficult than the single constraint version. Freville and Plateau (2004) designed an efficient preprocessing procedure for large-scale instances. The algorithm provides sharp lower and upper bounds on the optimal value, and also a tighter equivalent representation by reducing the continuous feasible set and by eliminating constraints

and variables. This scheme was shown to be very effective through a lot of computational experiments with test problems of the literature and large-scale randomly generated instances.

The binary quadratic knapsack problem maximizes a quadratic objective function subject to a linear capacity constraint. Due to its simple structure and challenging difficulty it has been studied intensively during the last two decades. Pisinger (2007) gave a survey of upper bounds presented in the literature, and showed the relative tightness of several of the bounds. Techniques for deriving the bounds include relaxation from upper planes, linearization, reformulation, Lagrangian relaxation, Lagrangian decomposition, and semi definite programming. A short overview of heuristics, reduction techniques, branch-and-bound algorithms and approximation results is given, followed by an overview of valid inequalities for the quadratic knapsack polytope. They concluded by an experimental study where the upper bounds presented are compared with respect to strength and computational effort.

Burkard and Pferschy (1995) dealt with parametric knapsack problems where the costs resp. weights are replaced by linear functions depending on a parameter t. The aim is to find the smallest parameter t^* such that the optimal solution value of the knapsack problem is equal to a prespecified solution value. For this inverse-parametric problem pseudo-polynomial algorithms were developed and search methods making use of the special properties of the parametric value function were constructed. Using computational experiments the behaviour of these algorithms are investigated and the favourable practical performance of different search methods exhibited.

The knapsack sharing problem (KSP) is formulated as an extension to the ordinary knapsack problem. The KSP is NP-hard. Yamada et al. (1998) presented a branch-and-bound algorithm and a binary search algorithm to solve this problem to optimality. These algorithms are implemented and computational experiments are carried out to analyse the behavior of the developed algorithms. As a result, they found that the binary search algorithm solves KSPs with up to 20 000 variables in less than a minute in their computing environment.

The objective of the multi-dimensional knapsack problem (MKP) is to find a subset of items with maximum value that satisfies a number of knapsack constraints. Solution methods for MKP,

both heuristic and exact, have been researched for several decades. Fleszar and Hindi (2009) introduced several fast and effective heuristics for MKP that are based on solving the LP relaxation of the problem. Improving procedures were proposed to strengthen the results of these heuristics. Additionally, the heuristics were run with appropriate deterministic or randomly generated constraints imposed on the linear relaxation that allow generating a number of good solutions. All algorithms were tested experimentally on a widely used set of benchmark problem instances to show that they compared favourably with the best-performing heuristics available in the literature.

The constrained compartmentalised 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 compartmentalised 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 maximise 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. Arenales and Marques (2007) modeled the problem as an integer non-linear optimisation problem and for which some heuristic methods were designed. Finally, computational experiments were given to analyse the methods.

Lin and Yao (2001) investigated knapsack problems, in which all of the weight coefficients are fuzzy numbers. The work was based on the assumption that each weight coefficient is imprecise due to the use of decimal truncation or rough estimation of the coefficients by the decision-maker. To deal with this kind of imprecise data, fuzzy sets provide a powerful tool to model and solve this problem. Their work was intended to extend the original knapsack problem into a more

generalized problem that would be useful in practical situations. As a result, their study showed that the fuzzy knapsack problem is an extension of the crisp knapsack problem, and that the crisp knapsack problem is a special case of the fuzzy knapsack problem.

Solving the knapsack problem with imprecise weight coefficients using genetic algorithms has been investigated. The work is based on the assumption that each weight coefficient is imprecise due to decimal truncation or coefficient rough estimation by the decision-maker. To deal with this kind of imprecise data, fuzzy sets provide a powerful tool to model and solve this problem. Lin (2008) investigated the possibility of using genetic algorithms in solving the fuzzy knapsack problem without defining membership functions for each imprecise weight coefficient. The proposed approach simulated a fuzzy number by distributing it into some partition points. A genetic algorithm was used to evolve the values in each partition point so that the final values represented the membership grade of a fuzzy number. The empirical results show that the proposed approach can obtain very good solutions within the given bound of each imprecise weight coefficient than the fuzzy knapsack approach. The fuzzy genetic algorithm concept approach is different, but gave better results than the traditional fuzzy approach.

The knapsack problem is believed to be one of the “easier” NP-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 showed that the knapsack problem still is hard to solve for these algorithms for a variety of new test problems. These problems were 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 challenged the dynamic programming algorithms while the other groups of problems were focused towards branch-and-bound algorithms. Numerous computational experiments with all recent state-of-the-art codes were used to show that (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.

The knapsack problem (KP) is generalized to the case where items are partially ordered through a set of precedence relations. As in ordinary KPs, each item is associated with profit and weight, the knapsack has a fixed capacity, and the problem is to determine the set of items to be packed in the knapsack. However, each item can be accepted only when all the preceding items have been included in the knapsack. The knapsack problem with these additional constraints is referred to as the precedence-constrained knapsack problem (PCKP). To solve PCKP exactly, Yamada and You (2007) presented a pegging approach, where the size of the original problem is reduced by applying the Lagrangian relaxation followed by a pegging test. Through this approach, they were able to solve PCKPs with thousands of items within a few minutes on an ordinary workstation.

Zhang and ong (2004) proposed a simple and useful method, the core of which is an efficient LP-based heuristic, for solving bi-objective 0–1 knapsack problems. Extensive computational experiments showed that the proposed method is able to generate a good approximation to the nondominated set very efficiently. They also suggested three qualitative criteria to evaluate such an approximation. In addition, the method can be extended to other problems having properties similar to the knapsack problem.

A promising solution approach called Meta-RaPS was presented by Moraga et al. (2005) for the 0-1 Multidimensional Knapsack Problem (0-1 MKP). Meta-RaPS construct feasible solutions at each iteration through the utilization of a priority rule used in a randomized fashion. Four different greedy priority rules are implemented within Meta-RaPS and compared. These rules differ in the way the corresponding pseudo-utility ratios for ranking variables are computed. In addition, two simple local search techniques within Meta-RaPS' improvement stage are implemented. The Meta-RaPS approach is tested on several established test sets, and the solution values are compared to both the optimal values and the results of other 0-1 MKP solution techniques. The Meta-RaPS approach outperformed many other solution methodologies in terms of differences from the optimal value and number of optimal solutions obtained. The advantage of the Meta-RaPS approach is that it is easy to understand and easy to implement, and it achieved good results.

Huttler and Mastrolilli (2006) addressed the classical knapsack problem and a variant in which an upper bound is imposed on the number of items that can be selected. We show that appropriate combinations of rounding techniques yield novel and more powerful ways of rounding. Moreover, they presented a linear-storage polynomial time approximation scheme (PTAS) and a fully polynomial time approximation scheme (FPTAS) that compute an approximate solution, of any fixed accuracy, in linear time. These linear complexity bounds give a substantial improvement of the best previously known polynomial bounds.

Gomes da Silva et al. (2007) dealt with the problem of inaccuracy of the solutions generated by meta-heuristic approaches for combinatorial optimization bi-criteria $\{0, 1\}$-knapsack problems. A hybrid approach which combines systematic and heuristic searches was proposed to reduce that inaccuracy in the context of a scatter search method. The components of this method were used to determine regions in the decision space to be systematically searched. Comparisons with small and medium size instances solved by exact methods were presented. Large size instances were also considered and the quality of the approximation was evaluated by taking into account the proximity to the upper frontier, devised by the linear relaxation, and the diversity of the solutions. Comparisons with other two well-known meta-heuristics were also performed. The results showed the effectiveness of the proposed approach for both small/medium and large size instances.

A critical event tabu search method which navigates both sides of the feasibility boundary has been shown effective for solving the multidimensional knapsack problem. In this paper, we apply the method to the multidimensional knapsack problem with generalized upper bound constraints. Li and Curry (2005) demonstrated the merits of using surrogate constraint information vs. a Lagrangian relaxation scheme as choice rules for the problem class. A constraint normalization method was presented to strengthen the surrogate constraint information and improve the computational results. The advantages of intensifying the search at critical solutions were also demonstrated.

Hanafi and freville (1998) described a new approach to tabu search (TS) based on strategic oscillation and surrogate constraint information that provides a balance between intensification

and diversification strategies. New rules needed to control the oscillation process are given for the 0 /1 multidimensional knapsack (0/1 MKP). Based on a portfolio of test problems from the literature, our method obtains solutions whose quality is at least as good as the best solutions obtained by previous methods, especially with large scale instances. These encouraging results confirm the efficiency of the tunneling concept coupled with surrogate information when resource constraints are present.

Pisinger (1995) presented a new branch-and-bound algorithm for the exact solution of the 0–1 Knapsack Problem is presented. The algorithm is based on solving an 'expanding core', which intially only contains the break item, but which is expanded each time the branch-and-bound algorithm reaches the border of the core. Computational experiments showed that most data instances are optimally solved without sorting or preprocessing a great majority of the items. The algorithm presented not only is shorter, but also faster and more stable than any other algorithm hitherto proposed.

The mean field theory approach to knapsack problems is extended to multiple knapsacks and generalized assignment problems with Potts mean field equations governing the dynamics. Numerical tests against "state of the art" conventional algorithms shows good performance for the mean field approach. The inherently parallelism of the mean field equations makes them suitable for direct implementations in microchips. Ohlsson and Pi (1997) demonstrated numerically that the performance is essentially not affected when only a limited number of bits is used in the mean field equations. Also, a hybrid algorithm with linear programming and mean field components is showed to further improve the performance for the difficult homogeneous N × M knapsack problem.

Rinnooy et al. (1993) proposed a class of generalized greedy algorithms is for the solution of the multi-knapsack problem. Items are selected according to decreasing ratios of their profit and a weighted sum of their requirement coefficients. The solution obtained depended on the choice of the weights. A geometrical representation of the method was given and the relation to the dual of the linear programming relaxation of multi-knapsack is exploited. They investigated the

complexity of computing a set of weights that gives the maximum greedy solution value. Finally, the heuristics were subjected to both a worst-case and a probabilistic performance analysis.

Optimization methods are being applied to engineering problem solving with increasing frequency as computer hardware and software improves. The configuration of an optimization algorithm can make a significant difference to the efficiency of the solution process. Realff et al., (1999) examined the use of one such optimization strategy, branch and bound, for the solution of the classic knapsack problem. It is shown that the best configuration of the algorithm can be data dependent and hence that an 'intelligent' optimization system will need to automatically configure itself with the control knowledge appropriate to the problems the user is solving. A two-step approach is taken to configuring the algorithm. First, an analytical learning method, explanation based learning is used to derive a provably correct dominance condition for the knapsack problem. Second, the algorithm is configured with and without the condition, and subjected to a rigorous statistical test of performance, on the user's data, to decide which configuration is the best.

Figuera et al. (2009) presented a generic labeling algorithm for finding all non-dominated outcomes of the multiple objective integer knapsack problem (MOIKP). The algorithm is based on solving the multiple objective shortest path problem on an underlying network. Algorithms for constructing four network models, all representing the MOIKP, were also presented. Each network is composed of layers and each network algorithm, working forward layer by layer, identifies the set of all permanent non-dominated labels for each layer. The effectiveness of the algorithms is supported with numerical results obtained for randomly generated problems for up to seven objectives while exact algorithms reported in the literature solve the multiple objective binary knapsack problem with up to three objectives. Extensions of the approach to other classes of problems including binary variables, bounded variables, multiple constraints, and time-dependent objective functions are possible.

The most efficient algorithms for solving the single-criterion {0,1}-knapsack problem are based on the core concept (i.e., based on a small number of relevant variables). But this concept is not used in problems with more than one criterion. Gomes da Silva et al. (2008) validated the

existence of such a set of variables in bi-criteria {0,1}-knapsack instances. Numerical experiments were performed on five types of {0,1}-knapsack instances. The results were presented for the supported and non-supported solutions as well as for the entire set of efficient solutions. A description of an approximate and an exact method was also presented.

There is a variation of the standard 0–1 knapsack problem, where the values of items differ under possible S scenarios. Taniguchi et al. (2008) introduced a kind of surrogate relaxation to derive upper and lower bounds quickly, and showed that, with this preprocessing, the similar pegging test can be applied to our problem. The reduced problem can be solved to optimality by the branch-and-bound algorithm. They made use of the surrogate variables to evaluate the upper bound at each branch-and-bound node very quickly by solving a continuous knapsack problem. Through numerical experiments they showed that the developed method finds upper and lower bounds of very high accuracy in a few seconds, and solves larger instances to optimality faster than the previously published algorithms.

Balev et al. (2008) presented a preprocessing procedure for the 0–1 multidimensional knapsack problem. First, a non-increasing sequence of upper bounds was generated by solving LP-relaxations. Then, a non-decreasing sequence of lower bounds is built using dynamic programming. The comparison of the two sequences allowed either to prove that the best feasible solution obtained is optimal, or to fix a subset of variables to their optimal values. In addition, a heuristic solution was obtained. Computational experiments with a set of large-scale instances show the efficiency of their reduction scheme. Particularly, it was shown that their approach allowed the reduction of the CPU time of a leading commercial software.

Balachandar and Kannan presented a heuristic to solve the 0/1 multi-constrained knapsack problem (0/1 MKP) which is NP-hard. In this heuristic the dominance property of the constraints is exploited to reduce the search space to find near optimal solutions of 0/1 MKP. This heuristic was tested for 10 benchmark problems of sizes up to 105 and for seven classical problems of sizes up to 500, taken from the literature and the results were compared with optimum solutions. Space and computational complexity of solving 0/1 MKP using this approach were also presented. The encouraging results especially for relatively large size test problems indicate that

this heuristic can successfully be used for finding good solutions for highly constrained NP-hard problems.

Florios et al. (2009) solved instances of the multi-objective multi-constraint (or multi-dimensional) knapsack problem (MOMCKP) from the literature, with three objective functions and three constraints. They used exact as well as approximate algorithms. The exact algorithm is a properly modified version of the multi-criteria branch and bound (MCBB) algorithm, which is further customized by suitable heuristics. Three branching heuristics and a more general purpose composite branching and construction heuristic were devised. Furthermore, the same problems are solved using standard multi-objective evolutionary algorithms (MOEA), namely, the SPEA2 and the NSGAII. The results from the exact case show that the branching heuristics greatly improve the performance of the MCBB algorithm, which becomes faster than the adaptive ε - constraint. Regarding the performance of the MOEA algorithms in the specific problems, SPEA2 outperforms NSGAII in the degree of approximation of the Pareto front, as measured by the coverage metric (especially for the largest instance).

While the 1980s focused on the solution of large sized "easy" knapsack problems (KPs), the 1990s brought several new algorithms, which were able to solve "hard" large sized instances. Martello et al. (2000) gave an overview of the recent techniques for solving hard KPs, with special emphasis on the addition of cardinality constraints, dynamic programming, and rudimentary divisibility. Computational results, comparing all recent algorithms, were presented.

Elhedhli (2005) considered a class of nonlinear knapsack problems with applications in service systems design and facility location problems with congestion. They provided two linearizations and their respective solution approaches. The first is solved directly using a commercial solver. The second is a piecewise linearization that is solved by a cutting plane method.

Caprara and Monaci (2004) addressed the two-dimensional Knapsack Problem (2KP), aimed at packing a maximum-profit subset of rectangles selected from a given set into another rectangle. They considered the natural relaxation of 2KP given by the one-dimensional KP with item weights equal to the rectangle areas, proving the worst-case performance of the associated upper

bound, and presented and compared computationally four exact algorithms based on the above relaxation, showing their effectiveness.

Abboud et al. (1997) presented an interactive procedure for the multi-objective multidimensional 0–1 knapsack problem that takes into consideration the incorporation of fuzzy goals of the decision maker, that is easy to use since it requires from the decision maker to handle only one parameter, namely, the aspiration level of each objective, and that is fast and can treat our problem as a usual 0–1 knapsack problem using already available software, namely, the primal effective gradient method, used primarily to solve the large-scale cases. To get some statistics on the behavior of the algorithm, a number of randomly generated simulation of problems was solved. From our numerical experience, it is possible to conclude that our proposed method is a worthwhile alternative to existing methods from a practical point of view.

Akinc (2006) addressed the formulation and solution of a variation of the classical binary knapsack problem. The variation that was addressed is termed the "fixed-charge knapsack problem", in which sub-sets of variables (activities) are associated with fixed costs. These costs may represent certain set-ups and/or preparations required for the associated sub-set of activities to be scheduled. Several potential real-world applications as well as problem extensions/generalizations were discussed. The efficient solution of the problem is facilitated by a standard branch-and-bound algorithm based on (1) a non-iterative, polynomial algorithm to solve the LP relaxation, (2) various heuristic procedures to obtain good candidate solutions by adjusting the LP solution, and (3) powerful rules to peg the variables. Computational experience shows that the suggested branch-and-bound algorithm shows excellent potential in the solution of a wide variety of large fixed-charge knapsack problems.

Index selection for relational databases is an important issue which has been researched quite extensively. In the literature, in index selection algorithms for relational databases, at most one index is considered as a candidate for each attribute of a relation. However, it is possible that more than one different type of indexes with different storage space requirements may be present as candidates for an attribute. Also, it may not be possible to eliminate locally all but one of the candidate indexes for an attribute due to different benefits and storage space requirements

associated with the candidates. Thus, the algorithms available in the literature for optimal index selection may not be used when there are multiple candidates for each attribute and there is a need for a global optimization algorithm in which at most one index can be selected from a set of candidate indexes for an attribute. The problem of index selection in the presence of multiple candidate indexes for each attribute (which we call the multiple choice index selection problem) has not been addressed in the literature. Gündem presented the multiple choice index selection problem, showed that it is NP-hard, and present an algorithm which gives an approximately optimal solution within a user specified error bound in a logarithmic time order.

In attempt to solve multi-objective problems, various mathematical and stochastic methods have been developed. The methods operate based on mathematical models while in most cases these models are drastically simplified imagine of real world problems. Gholamian et al. (2007) in their study, used a hybrid intelligent system i instead of mathematical models. The main core of the system is fuzzy rule base which maps decision space (Z) to solution space (X). The system is designed on non-inferior region and gives a big picture of this region in the pattern of fuzzy rules. Since some solutions may be infeasible; then specified feed forward neural network is used to obtain non-inferior solutions in an exterior movement. In addition, numerical examples of well-known NP-hard problems (i.e. multi-objective traveling salesman problem and multi-objective knapsack problem) were provided to clarify the accuracy of developed system.

Lokketangen and Glover (1998) described a tabu search (TS) approach for solving general zero-one mixed integer programming (MIP) problems that exploits the extreme point property of zero-one solutions. Specialized choice rules and aspiration criteria were identified for the problems, expressed as functions of integer infeasibility measures and objective function values. The first-level TS mechanisms were then extended with advanced level strategies and learning. They also look at probabilistic measures in this framework, and examine how the learning tool Target Analysis (TA) can be applied to identify better control structures and decision rules. Computational results are reported on a portfolio of multi-constraint knapsack problems. Their approach was designed to solve thoroughly general 0/1 MIP problems and thus contains no problem domain specific knowledge, yet it obtained solutions for the multi-constraint knapsack

problem whose quality rivaled, and in some cases surpassed, the best solutions obtained by special purpose methods that had been created to exploit the special structure of these problems.

Aissi et al. (2007) investigated, for the first time in the literature, the approximation of min–max (regret) versions of classical problems like shortest path, minimum spanning tree, and knapsack. For a constant number of scenarios, they established fully polynomial-time approximation schemes for the min–max versions of these problems, using relationships between multi-objective and min–max optimization. Using dynamic programming and classical trimming techniques, they construct a fully polynomial-time approximation scheme for min–max regret shortest path. They also established a fully polynomial-time approximation scheme for min–max regret spanning tree and prove that min–max regret knapsack is not at all approximable. For a non-constant number of scenarios, in which case min–max and min–max regret versions of polynomial-time solvable problems usually become strongly NP-hard, non-approximability results were provided for min–max (regret) versions of shortest path and spanning tree.

Jan et al. (2006) considered Web content adaptation with a bandwidth constraint for server-based adaptive Web systems. The problem can be stated as follows: Given a Web page P consisting of n component items $d_1, d_2, \ldots, d_n$ and each of the component items d_i having J_i versions $d_{i1}, d_{i2}, \ldots, d_{iJi}$, for each component item d_i select one of its versions to compose the Web page such that the fidelity function is maximized subject to the bandwidth constraint. They formulated this problem as a linear multi-choice knapsack problem (LMCKP) and transformed the LMCKP into a knapsack problem (KP) and then presented a dynamic programming method to solve the KP. A numerical example illustrated the method and showed its effectiveness.

Bortfeldt and Gehring (2001) presented a hybrid genetic algorithm (GA) for the container loading problem with boxes of different sizes and a single container for loading. Generated stowage plans include several vertical layers each containing several boxes. Within the procedure, stowage plans were represented by complex data structures closely related to the problem. To generate offspring, specific genetic operators were used that are based on an integrated greedy heuristic. The process takes several practical constraints into account. Extensive test calculations including procedures from other authors vouch for the good performance of the GA, above all for problems with strongly heterogeneous boxes.

Harper et al. (2001) presented a genetic algorithm as an aid for project assignment. The assignment problem illustrated concerns the allocation of projects to students. Students have to choose from a list of possible projects, indicating their preferred choices in advance. Inevitably, some of the more popular projects become 'over-subscribed' and assignment becomes a complex problem. The developed algorithm has compared well to an optimal integer programming approach. One clear advantage of the genetic algorithm is that, by its very nature, we are able to produce a number of feasible project assignments, thus facilitating discussion on the merits of various allocations and supporting multi-objective decision making.

Devyaterikova et al. (2009) presented discrete production planning problem which may be formulated as the multidimensional knapsack problem is considered, while resource quantities of the problem are supposed to be given as intervals. An approach for solving this problem based on using its relaxation set is suggested. Some *L*-class enumeration algorithms for the problem are described. Results of computational experiments were presented.

Dynamic programming is one of the most powerful Chen et al. (1990) presented pipeline architectures for the dynamic programming algorithms for the knapsack problems. They enabled them to achieve an optimal speedup using processor arrays, queues, and memory modules. The processor arrays can be regarded as pipelines where the dynamic programming algorithms are implemented through pipelining.

Making the provision of services QoS-aware is to the advantage of both clients and providers in the e-business domain. Tsesmetzis et al. (2008) studied the problem of providers that receive multiple concurrent requests for services demonstrating different QoS properties. It introduced the "Selective Multiple Choice Knapsack Problem" that aims to identify the services, which should be delivered in order to maximise the provider's profit, subject to maximum bandwidth constraints. This problem was solved by a proposed algorithm that has been empirically evaluated via numerous experiments.

Ahmed et al. (1987) considered the problem of selecting a set of projects from a large number of available projects such that at least some specified levels of benefits of various types are realized at a minimum cost. This problem was formulated in terms of the well-known 0–1 multi-dimensional knapsack problem, a special case of the general integer programming problems. In view of the NP-completeness of these problems, they proposed a polynomially bounded and efficient heuristic algorithm for its solution. The proposed algorithm proceeds as follows: an initial selection is found by prioritizing the projects according to a computed *discard index*. This initial selection set is then altered to reduce total costs by using project exchange operations. Computational results indicated that the proposed algorithm is quite effective in finding optimal or near optimal solutions.

Golenko-Ginzburg and Gonik (1997) presented a newly developed resource constrained scheduling model for a PERT type project. Several non-consumable activity related resources, such as machines or manpower, are imbedded in the model. Each activity in a project requires resources of various types with fixed capacities. Each type of resource is in limited supply with a resource limit that is fixed at the same level throughout the project duration. For each activity, its duration is a random variable with given density function. The problem is to determine starting time values S_{ij} for each activity (i,j) entering the project, i.e., the timing of feeding-in resources for that activity. Values S_{ij} are not calculated beforehand and are random values conditional on our decisions. The model's objective was to minimize the expected project duration. Determination of values S_{ij} was carried out at decision points when at least one activity is ready to be operated and there are free available resources. If, at a certain point of time, more than one activity is ready to be operated but the available amount of resources is limited, a competition among the activities is carried out in order to choose those activities which can be supplied by the resources and which have to be operated first. They suggested carrying out the competition by solving a zero-one integer programming problem to maximize the total contribution of the accepted activities to the expected project duration. For each activity, its contribution is the product of the average duration of the activity and its probability of being on the critical path in the course of the project's realization. Those probability values were calculated via simulation. Solving a zero-one integer programming problem at each decision point resulted in the following policy: the project management takes all measures to first operate those activities that, being

realized, have the greatest effect of decreasing the expected project duration. Only afterwards, does the management take care of other activities. A heuristic algorithm for resource constrained project scheduling was developed.

Project selection problem is an incessant problem, which every organization face. It, in fact, plays a key role in prosperity of the company. Meta-heuristic methods are the well-reputed methods which have been employed to solve a variety of multi-objective problems forming the real world problems. Ghorbani and Rabbani (2009) studied a new multi-objective algorithm for project selection problem. Two objective functions were considered to maximize total expected benefit of selected projects and minimize the summation of the absolute variation of allotted resource between each successive time periods. A meta-heuristic multi-objective was proposed to obtain diverse locally non-dominated solutions. The proposed algorithm was compared, based on some prominent metrics, with a well-known genetic algorithm, i.e. NSGA-II. The computational results showed the superiority of the proposed algorithm in comparison with NSGA-II.

A single machine scheduling problem in which the machine experiences the effects of learning of fatigue as it continues to work and the jobs have due dates and are subject to penalties if they are not completed on time. Because of the effects of learning or fatigue, the performance rate of the machine varies over time. As a result, the processing time of a job depends on its work content as well as the total work content of the jobs completed prior to its loading. Dondeti and Mohanty (1998) proved that even when the machine works at a variable rate, the pair-wise interchange of jobs minimizes the maximum tardiness and a simple modification to the well-known Moore-Hodgson's algorithm yields the minimum number of tardy jobs. Further, they formulated the problem of minimizing the total penalty for tardy jobs as a 0–1 knapsack problem with nested constraints, and solve it by using dynamic programming recursion as well as the maximum-weighted network path algorithm. Then they combined these two techniques and solve the 0–1 knapsack problem, by inducing a nested constraint structure and constructing a network with fewer nodes.

Carlo Vercellis (1994) described a Lagragean decomposition technique for solving multi-project planning problems with resource constraints and alternative modes of performing each activity in the projects. The decomposition can be useful in several ways: from one side, it provided bounds on the optimum, so that the quality of approximate solutions can be evaluated. Furthermore, in the context of branch-and-bound algorithms, it can be used for more effective fathoming of the tree nodes. Finally, in the modelling perspective, the La grangean optimal multipliers can provide insights to project managers as prices for assigning the resources to different projects.

An important class of combinational optimization problems are the Multidimensional 0/1 Knapsacks and various heuristic and exact methods have been devised to solve them. Among these, Genetic Algorithms have emerged as a powerful new search paradigms. Hoff et al showed how a proper selection of parameters and search mechanisms lead to an implementation of Genetic Algorithms that yields high quality solutions. The methods were tested on a portfolio of 0/1 multidimensional knapsack problems from literature and a minimum of domain-specific knowledge is used to guide the search process. The quality of the produced results rivals and in some cases surpasses the best solutions obtained by special-purpose methods that have been created to exploit the special structure of these problems

Fubin and Ru (2002) presented a simulated annealing (SA) algorithm for the o/1 multidimensional knapsack problem. Problem-specific knowledge is incorporated in the algorithm description and evaluation of parameters. In order to look into the performance of finite-time implementation of SA Computational results showed that SA performs much better than a genetic algorithm in term of solution time, whilst requiring only a modest loss of solution quality

Standard heuristics in operations research (such as greedy, tabu search and simulated annealing) work on improving a single current solution. Population heuristics use a number of current solutions and combine them together to generate new solutions. Heuristic algorithms encountered in the literature that can be generically be classified as population heuristics include genetic algorithms, hybrid genetic algorithms, memetic algorithms, scatter-search algorithms and bionomic algorithms. Beasley (2002) discussed the basic features of population heuristics and

provide practical advice about their effective use for solving operations research problems including the knapsack problems..

Lin and Wei (2001) proposed an efficient linear search algorithm for resolving 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 by Yang (1992) were tested and compared with the revised approach. Our results demonstrate that the approach proposed herein outperforms previous works in terms of producing a small candidate set while retaining most of the information on optimal

The transposition mechanism, widely studied in previous publications, showed that when used instead of standard crossover operators allows the genetic algorithm to achieve better solutions. Nevertheless, all the studies made concerning this mechanism always focused the domain of function optimization. Simoes and Costa (2001) presented an empirical study that compares the performances of the transposition-based Genetic Algorithm (GA) and the classical GA for solving the 0/1 knapsack problem. The obtained results showed that, just like in the domain of the function optimization, transposition is always superior to crossover.

A method of determining allocations in a business operation to maximize profit includes: collecting profit data for a plurality of classes in the business operation, where each class includes an allocation having a cost function and each allocation belongs to the group consisting of physical allocations and economic allocations; determining profit functions for the allocations from the profit data; formulating a Multiple Choice Knapsack Problem to maximize profit from the profit functions, the cost functions, and a cost constraint; and solving the Multiple choice Knapsack Problem to determine values for the allocations.(European Patent Application EP1350203)

The Knapsack Sharing Problem (KSP) is an NP-Hard combinatorial optimization problem, admitted in numerous real world applications. In the KSP, we have a knapsack of capacity c and a set of n objects, namely {\cal N}, where each object $j, j = 1, \ldots, n,$ is associated with a profit $p_j$and a weight w_j. The set of objects {\cal N} is composed of m different classes of objects. The aim is to determine a subset of objects to be included in the knapsack that realizes a max-min value over all classes. Hifi et al. (2002) solved the Knapsack Sharing Problem (KSP) using an

approximate solution method based upon tabu search. First, they described a simple local search in which a depth parameter and a tabu list were used. Next, they enhanced the algorithm by introducing some intensifying and diversifying strategies. The two versions of the algorithm yielded satisfactory results within reasonable computational time. Extensive computational testing on problem instances taken from the literature showed the effectiveness of the proposed approach.

CHAPTER THREE

TYPES OF KNAPSACK PROBLEMS AND SOLUTION METHODS

Because of their wide range of applicability, knapsack problems have known a large number of variations such as: single and multiple-constrained knapsacks, knapsacks 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.

3.1.1 The Single 0-1 Knapsack Problem

This is a 0-1 knapsack problem, pure integer programming with single constraint which forms a very important class of integer programming**.**

The 0-1 Knapsack Problem (KP) can be mathematically formulated through the following integer linear programming.

$$\text{Maximize} \sum_{j=1}^{n} P_j x_j$$

$$\text{Subject to} = \sum_{j=1}^{n} (w_j x_j) \leq c$$

$$x_j = 0 \text{ or } 1, \; j = 1, \ldots, n$$

3.1.2 The Subset Sum Knapsack problem

The particular case of the 0-1 knapsack problem arising when $p_j = w_j \; (j = 1,\ldots,n)$ as frequently occurs in practical applications. The problem is to find a subset of weights whose sum is closest to, without exceeding, the capacity, i.e.

Maximize $z = \sum_{j=1}^{n} (w_j x_j)$

Subject to $\sum_{j=1}^{n} (w_j x_j) \leq c$

$x_j = 0$ or 1, $\quad j = 1, \ldots, n$

This generally referred to as the Subset-Sum Problem.

3.1.3 The Change-Making Problem

A very particular bounded knapsack problem is considered arising when $p = 1$, $j = 1, \ldots \ldots, n$ and in the capacity constraint, we impose equality instead of inequality. This gives

Maximize $z = \sum_{j=1}^{n} (x_j)$

$Subject\ to \sum_{j=1}^{n} (w_j x_j) = b$

$x_j = 0$ or 1, $j = 1, \ldots \ldots, n$

This is usually called the Change-Making Problem, since it recalls the situation of a cashier having to assemble a given change c using the maximum (or minimum) number of coins.

3.1.4 Multiple Knapsack Problems

An important generalization of the 0-1 knapsack problem is the 0-1 Multiple knapsack problem arising when m containers, of given capacities c_i, $(i = 1, \dots, m)$ are available. By introducing binary variablesx_{ij}, taking value 1 if item j is selected for the container i, and value 0 otherwise, we obtain the formulation

$$\text{Maximize } z = \sum_{i=1}^{n} \sum_{j=1}^{n} (p_j x_{ij})$$

$$\text{Subject to } = \sum_{j=1}^{n} (w_{ij} x_j) \leq c_i$$

$$\sum_{j=1}^{n} x_{ij} \leq 1$$

$$x_j = 0 \text{ or } 1, \quad i = 1, \dots, n \quad j = 1, \dots, n$$

The generalization arising when the item set is partitioned into subsets and the additional constraint is imposed that at most one item per subset is selected is called the Multiple-Choice Knapsack Problem. The multi choice knapsack problem is defined as in knapsack problem with additional disjoint multiple choice constraint. The general description of the problem as given as follows: There is one knapsack with limited capacity. Objects to be packed in the knapsack are classified into multiple mutually exclusive classes. Within each class, there are several different items. The problem is to select some items from each class so as to minimize the total cost while the total size of the items does not exceed the limited capacity of the knapsack. This problem is of a generalized carryout problem and is NP-hard.

3.1.5 Multi-dimensional Knapsack problem

The multi-constraint is defined as a KP with a set of constraints such as weight, size, reliability etc. also called multi-dimensional knapsack problem.

The problem can be generalized by assuming that for each j $(j = 1, \dots, n), bj$ items of profit pj and weight wj are available $(bj \leq c/wj)$; thus we obtain the Bounded Knapsack problem,

3.2 Data Modeling

The 0-1 Knapsack Problem (KP) can be mathematically formulated through the following integer linear programming.

$$\text{Maximize} \sum_{j=1}^{n} P_j x_j$$

$$\text{Subject to} = \sum_{j=1}^{n} (w_j x_j) \leq c$$

$x_j = 0$ or $1, \quad j = 1, \dots, n$

3.2.1 Methods for solving Knapsack problems.

There are two basic methods for solving the 0-1 knapsack problems *(KP):* Theses are Branch-and-Bound and dynamic programming methods. However the use of meta-heuristics including Genetic algorithm, Tabu search and Simulated annealing have been used to solve large scale problems.

3.2.1 The Branch and Bound Method

Branch and Bound is a class of exact algorithms for various optimization problems, especially integer programming problems and combinatorial optimization problems (COP). It partitions the solution space into smaller subproblems that can be solved independently (branching).Bounding discards subproblems that cannot contain the optimal solution, thus decreasing thesize of the solution space. Branch and Bound was first proposed by Land and Doig in 1960 for solving integer programs.

Given a maximization problem

- a Branch and Bound algorithm iteratively partitions the solution space S, for example by branching on binary variables - fixing one of them to 0 in one branch and to 1 in the other branch.
- For each subproblem an upper bound on the objective value is calculated. The upper bound is guaranteed to be equal to or greater than the optimal solution for this subproblem.
- When a feasible solution (i.e., no fractional variables remaining) is found, all subproblems whose upper bounds are lower than this solution's objective value can be discarded.
- The best known feasible solution represents a lower bound for all subproblems, and only subproblems with an upper bound greater than the global lower bound have to be considered.

Discarding a subproblem is called fathoming or pruning. Upper bounds for a subproblem can be obtained by relaxing the subproblem, thus they are often obtained by optimizing the subproblem's LP relaxation.

The branch and bound method

Assume that the variables have been ordered such that

$$p_1 / a_1 \geq p_2 / a_2 \geq p_n / a_n$$

Let *s* be the maximum index *k* such that

$$\sum_{j=1}^{n}(a_j) \leq b$$

The following theorem due to Dantzig is shown below

The optimal solution to the continuous relaxation of KP is

$w_j = 1, \quad j = 1, \ldots \ldots, s$

$w_j = 0, \quad j = s, \ldots \ldots, n$

$$w_{s+1} = b - \sum_{j=1}^{n}\left(\frac{a_j}{a_{s+1}}\right)$$

If p_j, $j = 1, \ldots \ldots, n$ are positive integers, then an upper bound of the optimal value of KP is given by

$$UB = \sum_{j=i}^{s} P_j + [(b - \sum_{j=i}^{s} a_j)\, p_s + I/a_s + i]$$

$$UB = \sum_{j=1}^{n}(P_j) + [\,(b - \sum_{j=1}^{n} a_j)p_{s+1} / a_{s+1}\,]$$

where [x] is the largest integer less than or equal to x

The following branch-and-bound method uses the depth-first search and finds an upper bound by using the above theorem.

Algorithm Branch-And-Bound Method For Knapsack problem

Step 1 (Initialization).

Set $p_{N+1} = 0, \quad a_{N+1} = \infty, f_{opt} = f = 0, W_{opt} = W = (0,\ldots,0)^T, \quad W = b, i = 1$

Step 2 (Test heuristic). If $a_i \leq W$, find the largest s such that $\sum_{j=i}^{s} a_j \leq W$,

set $z = \sum_{j=i}^{s} p_j + \frac{(W - \sum_{j=i}^{s} a_j)p_{s+1}}{a_{s+1}}$. If $a_i > W$, set $s = i - 1$

and $z = Wp_s/a_s$. If $f_{opt} \geq [z] + f$, go to step 5.

Step 3 (New feasible solution). If $a_i \leq W$ and $i \leq N$, set $W := W - a_i$,

$f := f + p_i, \quad w_i = 1, i := i + 1$, repeat Step 3; otherwise, if $i \leq N$, set

$w_i = 0, i := i + 1$, if $i < N$, go to Step 2; if $i = N$, repeat Step 3;

if $i > N$, go to Step 4.

Step 4 (updating incumbent). If $f_{opt} < f$, set $f_{opt} = f, W_{opt} = W$. Set $i = N$,

if $W_N = 1$, set $W := W + a_N, \quad f := f - p_N, \quad W_N = 0$

Step 5 (Backtracking). Find the largest $k < i$ such that $W_k = 1$. If there is no such a k, stop and the current W_{opt} is the optimal solution. Otherwise, set

$W := W + a_k, f := f - p_k, \quad W_k = 0, \quad i = k + 1$ and go to step 2

3.2.2 Dynamic Programming Method

Dynamic Programming approach is applicable to (KP) if certain integrality conditions of the coefficients hold. We first assume that the coefficients a_j, $j = 1, \ldots\ldots, n$ are positive integers. *For each* $m = 1, \ldots, n$ *and* $z = 1, \ldots., b,$ *define*

$$Pm(z) = \max\{\sum_{j=1}^{m} p_j w_j \;/\; \sum_{j=1}^{m} a_j w_j \leq z, \quad (w_1, \ldots, w_m) \in \{0,1\}^m\}$$

The recursive equation at the m − th stage is

$p_m(z) = p_{m-1}(z), \quad 0 \leq z < a_m$

$\max\{ p_{m-1}(z), \quad p_{m-1}(z - a_m) + p_m\}, \quad a_m \leq z \leq b$

with the initial condition:

$p_1(z) = 0, \quad 0 \leq z < a_1$

$p_1 \quad a_1 \leq z \leq b$

Under the condition that a_j, $(j = 1, \ldots, N)$ are positive integers, a dynamic programming algorithm construct a table of dimension $N * (b + 1)$ and calculates the entries $p_m(z)$, $(m = 1, \ldots., N, z = 0, \ldots., b)$ in a bottom-up fashion. An optimal solution can be found by backtracking through the table once the optimal value $P_N(b)$ is obtained. The complexity of this dynamic programming is O (Nb).

3.2.3 Heuristic Scheme

A heuristic scheme that may be used to solve knapsack problems instead of branch and bound could be 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 value $f(S_0)$

Step 4: Obtain a new solution S_1 by flip operation and check for feasibility, continue flip operation until the solution S_1 so obtained is feasible. Compute the objective function value $f(S_1)$.

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

else retain S_0 and discard S_1

Step 5: Repeat step 3 for all feasible solutions

Step 6: Stop for not improving solution over a number of iterations

3.2.4 Simulated Annealing

Simulated annealing is a local search algorithm capable of escaping from local optima. Its case of implementation, convergence properties and its capability of escaping from local optima has made it a popular algorithm over the past decades. 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. i.e. its minimum lattice energy state and thus is free of crystal effects. Simulated annealing mimics this type of thermodynamic behavior in searching for global optima for discrete optimization problems (DOP).

At each iteration of simulated annealing, algorithm applied to a DOP, 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 provides a means to escape local optima by allowing 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 which means that the algorithm is asymptotically convergent.

To formally describe simulated annealing algorithm for KP, some definitions are needed. Let Ω be the solution space: define $\eta(\omega)$ to be the neighborhood function for $w \in \Omega$. Simulated annealing starts with an initial solution $\omega \in \Omega$. A neighborhood solution $\omega^{1} \in \eta(\omega)$ 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 $\omega \in \Omega$ to a candidate solution $\omega i \in \eta(\omega)$ 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 survey, 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. Simulated annealing algorithm with static cooling schedule (Kirkpatrick et al. 1983) for KP is outlined in pseudo-code.

1. Select an initial solution $\omega = (\varkappa_1, \ldots, \varkappa_n) \in \Omega$; an initial temperature $t = t_0$;
2. control parameter value α; final temperature e; a repetition schedule, M that defines the number of iterations executed at each temperature;
3. Incumbent solution $\leftarrow f(\omega)$;
4. Repeat;
5. Set repetition counter $m = 0$;
6. Repeat;
7. Select an integer i from the set $\{1,2, \ldots, n\}$ randomly:
8. If $x_i = 0$, pick up item i, i.e. set $x_i = 1$, obtain new solution $\omega 1$ then
9. while solution $\omega 1$ is infeasible, do
10. drop another item from ω randomly; denote the new solution as $\omega 1$
11. let $\Delta = f(\omega 1) - f(\omega)$
12. while $\Delta \geq 0$ or Random (0,1) < $e^{\Delta/t}$ do $\omega \leftarrow \omega 1$
13. Else
14. drop item i and pick another item randomly, get new solution $\omega 1$
15. let $\Delta = f(\omega 1) - f(\omega)$
16. while $\Delta \geq 0$ or Random (0,1) $< e^{\Delta/t}$ do $\omega \leftarrow \omega 1$
17. End If
18. If incumbent solution $< f(\omega)$, Incumbent solution $\leftarrow f(\omega)$
19. $m = m + 1$;
20. Until $m = M$
21. set $t = a * t$;
22. Until $t < e$

A set of parameters needs to be specified that govern the convergence of the algorithm, i.e. initial temperature to, temperature control parameter α, final temperature e, and Markov chain length M, in order to study the finite-time performance of simulated annealing algorithm. Here t_o should be the maximal difference in cost between any two neighboring solutions

3.2.4 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. During 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 (i.e. children) 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.

The basic steps of a simple GA are shown below

Step 1: Generate an initial population

Step 2: Evaluate fitness of individuals in the population

Step 3: repeat

a. Select individuals from the population to be parents
b. Recombine (mate) parents to produce children
c. Mutate the children Evaluate fitness of the children
d. Replace some or all of the population by the children

until

Step 4: you decide to stop whereupon report the best solution encountered

What Does An Individual In Our GA World Look Like?

In the real world we know what individuals look like. In the GA world, what individuals look like (their representation or chromosome) is our choice.

In our GA world for the KP we shall choose individuals to be n bit 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 $x2 = x6 = 1\ and\ x1 = x3 = x4 = x5 = x7 = 0$

step 2 Evaluation of fitness

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

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

Step 3a: Selection of individuals for as parents

In the real world, individuals are independent beings who for their own reasons decide to become parents. But in the GA world we have to make a choice as to who will become a parent.

In ther GA world for the 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 3b: Recombine (Mate) Parents To Produce Children

In the real world we know parents combine to have children.

In the GA world for the 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

Other ways are (briefly) outlined below.

One-Point Crossover

In one-point crossover we randomly select a pint 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 0 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

Observe that in the on-point crossover example presented above we would 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 restricts the cut point to ensure that the children are different from the parents. That is easily done, simply restrict 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)

- fusion, as uniform crossover except that bits are taken from the parents with probabilities proportional to their fitness;
- 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 taking alternate segments from each parent

Indeed, any way of combining two bit strings together could be used to produce children from two parents. Note here however, 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. (i.e. characteristics common to parents are passed to the children).

Step 3c: 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 example we might randomly select it 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 3d: Infeasibility

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

There are number of strategies for dealing with constraints and infeasible solutions in Gas 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 (Beasley and Chu, 1996), 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 the KP and we illustrate this below

Strategies for Dealing with Constraints and Infeasible Solutions in Genetic Algorithm

Strategy	Description
1	To use a representation that automatically ensures that all solutions are feasible
2	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
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 a feasible solution is trivial, for example,

repeat until solution feasible:

$$set\ x_j = 0$$

Population Replacement

We will use a steady-state population replacement strategy. With this strategy 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 delete ("kill").

A logical approach is to kill the member of the population with the worst fitness.

Modified GA

Generate an initial population

- Evaluate fitness of individuals in the population

repeat

- Select individuals from thee population to be parents

- Recombine (mate) parents to produce children
- Mutate the children
- Make the children feasible using the heuristic operator
- Evaluate fitness of the children
- Replace some or all of the population by the children

Until

you decide to stop whereupon report the best solution encountered

The difference between this description and the algorithm description given previously is the insertion of a step making the (mutated) children feasible.

Computational Considerations

The GA presented above (with a few modifications) produces results that are superior in quality to other leading heuristic (which are mostly based on tabu search) for the KP (Chu and Beasley, 1998). However, as already mentioned, that GA is much slower than other heuristics. Hence, we have the trade-off we often see in OR between quality of solution and computer time consumed.

GAs often require the production and evaluation of many different children. However, GAs are capable of generating high-quality solutions to many problems within reasonable computation times. (Beasley and Chu, 1996; Chu and Beasley, 1997, 1998; Chang et al., 2000; Beasley et al., 1999)

CHAPTER 4

DATA COLLECTION AND ANALYSIS

4.1. Data Collection

The study area is the selection of adverts at GTV. GTV is a state owned Television station which depends to the greater extent on government subvention. GBC is however mandated to generate revenue to supplement the government subvention. To this end GTV has various ways of generating additional income. These include sponsorship of programmes , social and funeral announcements, advertisements among others. However this research focused on advertisements which are slotted in the programmes schedules (appendix A) prepared quarterly to generate additional income to sustain the operations of the TV station.

Spots for adverts are categorized into the following

- Prime time news (19.00 hrs GMT)
- News adjacencies (five minutes before and after news at 12.00, 14.00, 19.00 and 22.30 hours GMT)
- Other News time (12.00, 14.00, 19.00, 22.30 hours GMT)
- Break in programmes (peak and off peak)

Each of the above categories has different rates attached as shown below .

The table 1 shows the various rates for the different categories of adverts at GTV. For example a Primetime News adverts for 15 seconds costs GH¢215 while for 60 seconds, the rate is GH¢525. the rates are high for Prime time News and news adjacencies. These are periods where most customers wants their adverts televised to reach a larger TV audience. The off peak rates are low compared with the peak periods.

Table 1: GTV Adverts Rate

	Rates in GHC			
Category/Time	**15 sec**	**30 sec**	**45 sec**	**60 sec**
Prime Time News(19hrs GMT)	215.00	375.00	562.00	750.00
News Adjacencies	130.00	250.00	375.00	500.00
Break in News	135.75	244.35	362.00	525.00
Break in Programmes				
a. Peak Time - Week Days	91.00	160.00	220.00	360.00
b. Peak Time – Week Ends/Holidays	70.00	120.00	164.00	271.00
c. Off Peak	45.00	61.00	120.00	177.00

Customers usually request for a number of spots for their adverts. The table 2 shows represents request received by GTV at Primetime News (19 hours GMT). Customer 1 requested for two(2) spots of adverts for fifteen (15) seconds each at prime time news. The cost of the two adverts is GH¢260 (i.e 130 + 130) as indicated in the value column. The weights of this advert is 30sec. Additionally customer number 5 requested 3 spots of 30 seconds each. ie. 90 sec(weight) with a cost of GH¢1125(value). The total time available for adverts at the prime time news is twenty (20) minutes (i.e 1200 seconds) but the total time requested is 1710 seconds.

Table 2: Prime Time News Adverts – 19:00 Hours GMT

Adverts No.	Time in sec (t)	No of spots requested(s)	Time requested(weight)	Cost(Value)
1	15	2	30	429
2	30	3	90	1125
3	45	1	45	562
4	15	1	15	214
5	30	3	90	1125
6	45	2	90	1124
7	60	1	60	750
8	30	2	60	750
9	45	2	90	1124
10	15	1	15	215
11	15	1	15	215
12	30	1	30	375
13	45	2	90	1124
14	15	2	30	429
15	30	3	90	1125
16	45	2	90	1124
17	30	3	90	1125
18	30	3	90	1125
19	45	2	90	1124
20	60	1	60	750
21	45	1	45	562
22	15	1	15	215
23	15	1	15	215
24	15	1	15	215
25	30	2	60	750
26	30	3	90	1125
27	15	2	30	429
28	60	1	60	750
29	30	3	90	1125
30	15	2	30	429
Total			1710	

Other customers opt for the News Adjacencies. this is five (5) minutes before and after the prime time news at 19.00 hours GMT. As shown in table 3, the total time available is 10 minutes (600 seconds) but the customers requested a total of 810 seconds

Table 3. Adverts for News Adjacencies -18:55 -19:00 and 20:00-20:05

Adverts No.	**Time requested (weight)**	**Cost GHC(Value)**
1	30	260
2	45	375
3	15	130
4	90	750
5	60	500
6	60	250
7	90	750
8	15	130
9	15	130
10	30	250
11	30	260
12	60	500
13	45	375
14	15	130
15	15	130
16	15	130
17	60	250
18	30	260
19	60	500
20	30	260
Total	**810**	

Table 4 shows the weights and the values for the adverts requested for the 22:30 news time. The total time available is 600 seconds but the customers requested 810 seconds.

Table 4. Selected adverts for Break in News at 22:30 Hours GMT

Adverts No.	Time requested (weight)	Cost GHC(Value)
1	30	150
2	45	200
3	15	75
4	90	400
5	60	290
6	60	270
8	15	75
9	15	75
10	30	135
11	30	150
12	60	290
13	45	200
14	15	75
15	15	75
16	15	75
17	60	270
18	30	150
19	60	290
20	30	150
Total	**720**	

Table 5. Break in programme adverts for Peak Time on Week Days

Adverts No.	Time requested (weight)	Cost GHC(Value)
1	15	91
2	15	91
3	30	160
4	90	440
5	30	182
6	90	480
7	90	440
8	90	480
9	60	320
10	15	91
11	15	91
12	15	91
13	60	320
14	90	480
15	30	182
16	60	360
17	90	480
18	30	182

4.2 Data Analysis

The Data collected for GTV was analyzed with the computer software developed in Visual Basic.Net 2008 using the heuristic scheme algorithm .

4.2.1 **Features of the Software**

The software allows the user to input data into the program in three ways as shown in the User interface below by the radio buttons.(codes attached: appendix B)

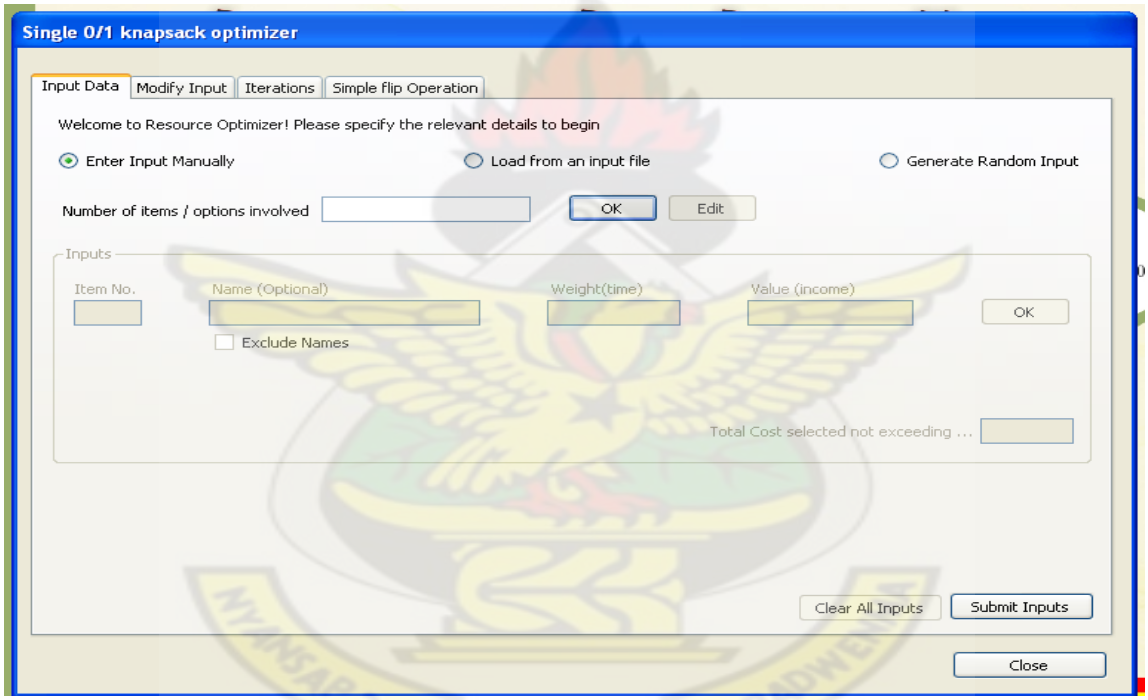
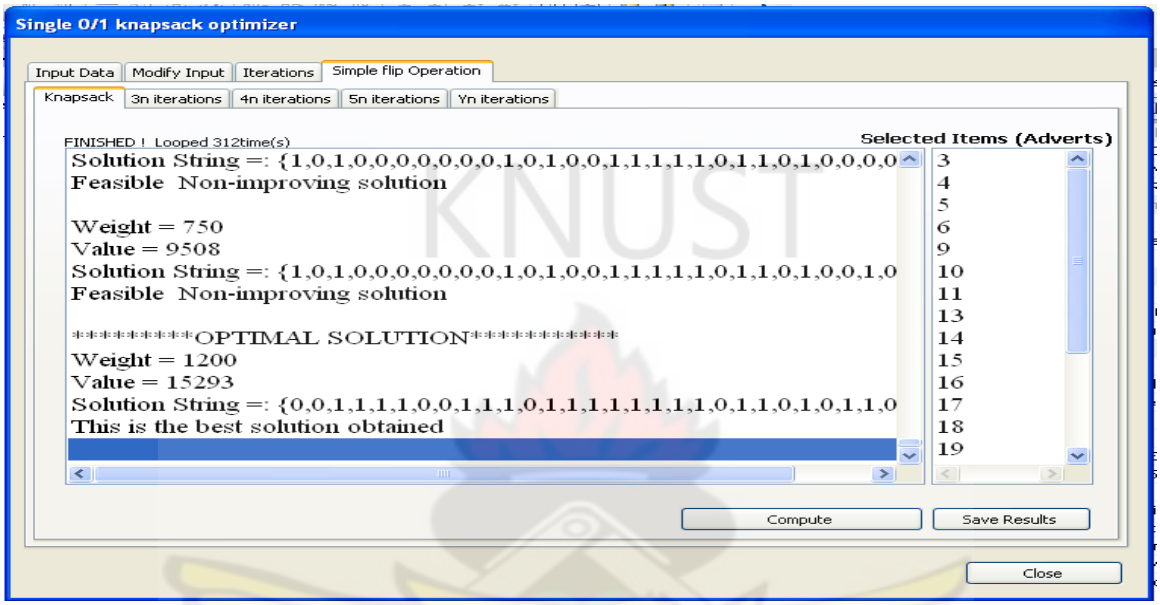


fig 1. User Interface for the Knapsack Optimizer

- The user can load an existing data already stored in the computer
- The user may type in the data directly into the textboxes
- For testing purposes the user can generate data automatically.

The programme generate an initial solution and shows all feasible solutions for the problem and selects the optimal solution. The optimal solution gives the solution string, the weight and the value. The selected adverts are the listed in a list box to the right as shown below.



4.2 .2 Results of the Analysis

Results for the analysis of data from a prime time news, news adjacencies, break in News and break in programme are shown below. The optimal selection these adverts yielded GHC 26,305.

From the table below nineteen (19) adverts were selected from the 30 requested to give an optimal value of GHC 15,157. The selection for the break in news , break in programme and a peak period yielded GHC 2,820, GHC 3,288, GHC 5,040 respectively. These are higher than as compared with the results of the arbitrary method used by GTV. Additionally, more adverts for each category of advertisement was selected as compared with the existing method of selection.

	No. of Adverts Requested	No. of Adverts Selected	Time Available in Second	Optimal Value (Amount)
Prime Time	30	19	1200	15,157
News Adjacencies	20	14	600	5,040
Break in News	20	14	600	2,820
Break in Programme	20	13	600	3,288
Total				26,305

The adverts selected are as follows

- Prime Time News - {3,6,8,9,12,13,15,16,17,18,19,21,23,24,25,27,28,29,30}
- News adjacencies - {1,2,4,57,8,910,12,13,15,16,19,20}
- Break in News - {1,4,5,6,9,10,11,12,13,14,15,16,17,18}
- Break in Programme – {1,2,3,5,6,7,9,11,12,13,14,16,18}

CHAPTER FIVE

CONCLUSION AND FUTURE WORK

We have described the TV adverts selection problem as a 0–1 knapsack optimization problem. Given that a 0–1 knapsack optimization problem is NP-hard, we used the simple heuristic scheme to solve the TV adverts problem.

The areas of our research was the use of the Knapsack problem for selecting adverts in critical situations such as the prime time news (19.00 hours GMT) and news adjacencies and other news time (12,14,22:30 hours GMT). However it can be applied to any situations where advertisers opt for the same limited spots such as sponsoring international football and other events which will attract many viewers.

For a typical day an amount of GHC26,305 was obtained from adverts selected for the four categories of adverts, which is far in excess of the arbitrary selection method used by GTV. This translates to GHC 2,367450 for a 3 month period.

The use of the software is systematic and transparent as compared with the arbitrary method. Higher returns can be achieved by GTV by the use of this software in their selection of adverts in the critical situations analysed.

Marketing Managers / Programme Producers may benefit from the proposed approach for selecting adverts to guarantee maximized profits for their TV stations.

In an event where management may have to include certain adverts for national interest these could be isolated before selecting the others to compete for the limited time slots.

The software can be used for any problem that can be modeled as a 0/1 knapsack problem.

Finally, we only considered single criteria for the selection of the TV adverts. Further research is needed for applying multi-criteria and multiple knapsack problems to the TV adverts selecting problem.

REFERENCES

1. Abboud, N. J., M. Sakawa, M. Inuiguchi (1997). A fuzzy programming approach to multiobjective multidimensional 0–1 knapsack problems. www.sciencedirect.com
2. Arnaud Freville, Gérard Plateau (2004). An efficient preprocessing procedure for the multidimensional 0–1 knapsack problem. www.sciencedirect.com
3. Arne Lokketangen, Fred Glover (1998). Solving zero-one mixed integer programming problems using Tabu search. www.sciencedirect.com
4. Beasley J. E., P. C. Chu (1996). A genetic Algorithm for the set covering problem. European Journal of Operations Research 94:392-404
5. Byungjun You, Takeo Yamada (2007). A pegging approach to the precedence-constrained knapsack problem. www.sciencedirect.com
6. Carlos Gomes da Silva, João Clímaco, José Rui Figueira (2008). Core problems in bi-criteria {0,1}-knapsack problems. www.sciencedirect.com
7. Chang, J.T., N. Meade, J. E. Beasley, Y.M. Sharaiha. (2000). Heuristics for cardinality constrained portfolio optimization. Comp. Operasions. Research. 27: 1271-1302
8. Chu, P. C., J. E. Beasley (1997). A genetic algorithm for generalized assignment problem. Computer Operations Research 24: 17-23
9. Chu, P. C., J. E. Beasley (1998a). A genetic algorithm for multidimentional knapsack problem. Journal Heuristics 4: 63-68
10. Chu, P. C., J. E. Beasley (1998b). Constraint handling in genetic algorithm: the set partitioning problem. Journal Heuristics 4: 323-357
11. David Pisinger (1995). An expanding-core algorithm for the exact 0–1 knapsack problem

12. David Pisinger (2005). Where are the hard knapsack problems? www.sciencedirect.com
13. David Pisinger (2007). The quadratic knapsack problem—a survey. www.sciencedirect.com
14. Devyaterikova, M.V., A.A. Kolokolov, A.P. Kolosov (2009). L-class enumeration algorithms for a discrete production planning problem with interval resource quantities
15. Dimitri Golenko-Ginzburg, Aharon Gonik (1997). Stochastic network project scheduling with non-consumable limited resources. www.sciencedirect.com
16. Eugénia Captivo, M., João Clímaco, José Figueira, Ernesto Martins, José Luis Santos, (2003). Solving bicriteria 0–1 knapsack problems using a labeling algorithm
17. Fabiano do Prado Marques, Marcos Nereu Arenales (2007). The constrained compartmentalised knapsack problem. www.sciencedirect.com
18. Feng-Tse Lin (2008).Solving the knapsack problem with imprecise weight coefficients using genetic algorithms. www.sciencedirect.com
19. Feng-Tse Lin, Jing-Shing Yao (2001). Using fuzzy numbers in knapsack problems
20. Fleszar, Khalil S. Hindi (2009). Fast, effective heuristics for the 0–1 multi-dimensional knapsack problem. www.sciencedirect.com
21. Freville, A., G. Plateau (1986). Heuristics and reduction methods for multiple constraints 0–1 linear programming problems. www.sciencedirect.com
22. Fumiaki Taniguchi, Takeo Yamada, Seiji Kataoka (2008). Heuristic and exact algorithms for the max–min optimization of the multi-scenario knapsack problem
23. Gen-Huey Chen, Maw-Sheng Chern, Jin-Hwang Jang (1990). Pipeline architectures for dynamic programming algorithms. www.sciencedirect.com

24. Gholamian, M.R., S.M.T. Fatemi Ghomi, M. Ghazanfari (2007). A hybrid system for multiobjective problems – A case study in NP-hard problems. www.sciencedirect.com
25. Ghorbani, S., M. Rabbani (2009). A new multi-objective algorithm for a project selection problem. www.sciencedirect.com
26. Gündem,T. I. (1999). Near optimal multiple choice index selection for relational databases. www.sciencedirect.com
27. José Rui Figueira, Gabriel Tavares, Margaret M. Wiecek (2009). Labeling algorithms for multiple objective integer knapsack problems
28. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P. (1983). Optimization by simulated annealing.
29. Kostas Florios, George Mavrotas, Danae Diakoulaki (2009). Solving multiobjective, multiconstraint knapsack problems using mathematical programming and evolutionary algorithms
30. Li, V. C., G. L. Curry (2005). Solving multidimensional knapsack problems with generalized upper bound constraints using critical event tabu search. www.sciencedirect.com
31. Mattias Ohlsson Hong Pi (1997). A Study of the Mean Field Approach to Knapsack Problems. www.sciencedirect.com
32. Monaldo Mastrolilli, Marcus Hutter (2006). Hybrid rounding techniques for knapsack problems. www.sciencedirect.com
33. Raja Balachandar, S., K. Kannan (2008). A new polynomial time algorithm for 0–1 multiple knapsack problem based on dominant principles. www.sciencedirect.com

34. Realff, M. J., P. H. Kvam, W. E. Taylor (1999) Combined analytical and empirical learning framework for branch and bound algorithms: the knapsack problem. www.sciencedirect.com
35. Reddy Dondeti, V., Bidhu B. Mohanty (1998). Impact of learning and fatigue factors on single machine scheduling with penalties for tardy jobs. www.sciencedirect.com
36. Reinaldo J. Moraga, Gail W. DePuy, Gary E. Whitehouse (2005). Meta-RaPS approach for the 0-1 Multidimensional Knapsack Problem. www.sciencedirect.com
37. Rinnooy Kan, A.H. G. L., Stougie, C. Vercellis (1993). A class of generalized greedy algorithms for the multi-knapsack problem. www.sciencedirect.com
38. Robert M. Nauss (1978). The 0–1 knapsack problem with multiple choice constraints.
39. Saïd Hanafi, Arnaud Freville (1998). An efficient tabu search approach for the 0–1 multidimensional knapsack problem. www.sciencedirect.com
40. Samir Elhedhli (2005). Exact solution of a class of nonlinear knapsack problems.
41. Samuel Eilon (1987). Application of the knapsack model for budgeting
42. Silvano Martello, David Pisinger, Paolo Toth (2000). New trends in exact algorithms for the 0–1 knapsack problem
43. Simoes, A., Costa, E. Transposition versus Crossover. An Emperical study. Proceedings of the genetic and evolutional computation conference (1999): pp 612-619
44. Simoes, A., Costa, E.: Using Genetic Algorithm with Asexual Transposition. Proceedings of the genetic and evolutional computation conference (2000): pp 323-330
45. Stefan Balev, Nicola Yanev, Arnaud Fréville, Rumen Andonov (2008). A dynamic programming based reduction procedure for the multidimensional 0–1 knapsack problem

46. Takeo Yamada, Mayumi Futakawa, Seiji Kataoka (1998). Some exact algorithms for the knapsack sharing problem. www.sciencedirect.com
47. Tao Zhong, Rhonda Young (2009). Multiple Choice Knapsack Problem: Example of planning choice in transportation. www.sciencedirect.com
48. Theodore S. Glickman, Stephen V. Allison (1973). Investment planning for irrigation development projects
49. Ulrich Pferschy, David Pisinger, Gerhard J. Woeginger (1997). Simple but efficient approaches for the collapsing knapsack problem. www.sciencedirect.com
50. Umit Akinc (2006). Approximate and exact algorithms for the fixed-charge knapsack problem.www.sciencedirect.com

APPENDIX A - GTV Programme Schedule July – September 2009

Time	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
5:00am	World Net						
5:30am	Voice Of Healing	Deeper Life	Ebenezer Miracle Worship Center	World Net	Pentecost Hour	Pranic Healing	World Net
6:00am	Apostolic Heritage	Breakfast Show					Catholic Digest
6:30am	Encounter With Truth						Restoration Hour
7:00am	Voice Of Inspiration						Life Matters Winners Church Gh
7:30am	Apostolic Voice						Winning Ways
7:55am	Programme Line-Up						Programme Line-Up
8:00am	Channel Of Hope						Breakfast Show
8:30am	Keepers Of Faith	Dwtv					
9:00am	In The House	Psi-Distance Learning	M'brasem	Psi-Distance Learning	Sports Lite	Asem Sebe	
9:30am							Barneys
10:00am	News Highlight						
10:05am	All Of Us	Psi-Distance Learning	Meet The Press	Psi-Distance Learning	Meet The Press/Straight Talk Africa	Asem Sebe	Children's Channel
10:30am	Gospel Trail						

11:00am	News Highlight						
11:05am	Documentary	Court Precision	Total Football	Wiase Ye Sum	Meet The Press(Repeat)	Health Talk	National Scince& Maths Quiz
11:30am	Paid Music			Regional Diaries(Repeat)		Kasa Mame	
12:00pm	News	Yese Yese		Primwell	African Movie	Family Movie	News
12:30pm	Cantata	In The House(Repeat		African Movie			Aware Pa
1:00pm							Sport Beat
1:30pm	What Do You Know	"What I Sell"				Soccer Icon	
2:00pm		**News**					Our Children Our Future
2:30pm	Enye Easy	Business Africa	Standpoint (Repeat)	Words Of Peace	Ads Forum	Bundesliga Kickoff	This Week
3:00pm	Stars Of The Future	Royal Whispers		Garage	W'adidiemu Te Sen	This Week In French	Next Level
3:30pm		Insurance &You	Kings&Queens		Afro Tv	Tech Express	Asenta Oba
4:00pm	News Highlight						
4:05pm	O Baby!	My Lovely Sam -Soon	Psi-Distance Learning	Paid Documentaries &Musicals	Psi-Distance Learning	Islam&Life	Traffic Warden/Epl
4:30pm	Borges Health Check			Crime Fighters		Movie Web	Chldren Of Today
5:00pm	News Highlight						
5:05pm	Complete Woman	Regional Diaries	Psi-Distance Learning	Environmatazz (Min.Of Information)	Psi-Distance Learning	Local Drama	Miss Ghana & Dance Championship
5:30pm	Maggie Food Moments	Jamin Reggae		Game Time(Charter			

				House)			
6:00pm	News Highlight						
6:05pm	Frees The Slaves	Adult Education In Akan	Adult Education In Ausa/Dagbani	Adult Education In Ga	Adult Education In Ewe	Adult Education In Nzema	On The Ball
6:30pm	M'asem	Soc/Funeral Ann./Line-Up					Inside Out
6:45pm		News In	News In Akan	News In Dagbani	News In Ga	News In Ewe	
7:00pm	**News/Businees/Weather**						
7:30pm	Talking Point	News/Business/Weather					
8:00pm		Sports Highlights	Mmaa Nkomo	Investment Digest	Possibility Forum	Standpoint	Rythms
8:30pm	Zain African Challenge	Sports Highlights	Mmaa Nkomo	Zain African Challenge	Business Advocate		
9:00pm	Hot Bench	Time With Nafti	It Takes Two	Secrets		Ghana's Pride	Wicked Games
9:300pm	Obra/Nsem Bi Sisi	Fortune Island	Stars Of The Future	Back Home Again	Faith Talk	Dr. Payne	
10:00pm		Pasion					Eve
10:30pm	**Late News**						
11:00pm	Power In His Presence	Healing Jesus Crusade	Guiness Football Africa	Way Of Life	"In Him Is Life"	Counselling Session	Primwells/African Movie
11:30pm	H.M. Films	Dilema	Documentary				African Movie
12:00pm	African Movie	Paid Musicals					
12:30am	DWTV						
1:00am	Close Down						

APPENDIX B - VISUAL BASIC.NET CODES FOR THE HEURISTIC SCHEME

```
Imports System

Imports System.IO

Imports System.Collections

Public Class main

    'global variable declarations

    Dim nInputs As Integer, upperLimit As Integer = 0

    Dim value(1000) As Integer, cost(1000) As Integer, names(1000) As String

    Dim newSolution As Solution, currentSolution As Solution

        'end global variable declarations

    ' all my user-defined classes, functions and subroutines

 Public Class Solution

  Public itsArray(1000) As Integer, solString As String

        Public profitFunction As Integer, costFunction As Integer

Public feasible As Boolean = False
```

```
Public Sub valueFunctions()
costFunction = 0 : profitFunction = 0 : solString = "{"
 For i = 1 To main.nInputs
 costFunction = costFunction + (main.cost(i) * itsArray(i))
profitFunction = profitFunction + (main.value(i) * itsArray(i))
            If i = main.nInputs Then
               solString = solString + itsArray(i).ToString + "}"
            Else
               solString = solString + itsArray(i).ToString + ","
            End If
         Next i
      End Sub

 Public Sub New(ByVal init As Solution)
         For i = 1 To main.nInputs
            itsArray(i) = init.itsArray(i)
         Next
```

```
        valueFunctions()
        feasible = init.feasible
    End Sub
    Public Sub New()
    End Sub
End Class
Public Sub flipOperation(ByVal aSolution As Solution)
    Dim randomIndex As Integer
    statusLabel.Text = "Flip any randomly chosen index in the solution ..."
    Randomize()
    randomIndex = CInt(((nInputs - 1) * Rnd()) + 1)
    aSolution.itsArray(randomIndex) = 1 - aSolution.itsArray(randomIndex)
    aSolution.valueFunctions()
    If aSolution.costFunction <= upperLimit Then
        aSolution.feasible = True
    Else
        aSolution.feasible = False
```

```
    End If
End Sub


Public Sub flipOperation(ByVal whatToFlip As Integer, ByVal aSolution As Solution)
    Dim randomIndex As Integer
    statusLabel.Text = "Searching for an index equal to " + whatToFlip.ToString + "  to flip ..."
    Do
        Randomize()
        randomIndex = CInt(((nInputs - 1) * Rnd()) + 1)
    Loop Until aSolution.itsArray(randomIndex) = whatToFlip
    aSolution.itsArray(randomIndex) = 1 - aSolution.itsArray(randomIndex)
    aSolution.valueFunctions()
    If aSolution.costFunction <= upperLimit Then
        aSolution.feasible = True
    Else
        aSolution.feasible = False
    End If
```

```
End Sub


Private Sub printOut(ByVal aSolution As main.Solution, ByVal lastString As String, ByRef thelistbox As ListBox)
    thelistbox.Items.Add("Weight = " + aSolution.costFunction.ToString)
    thelistbox.Items.Add("Value = " + aSolution.profitFunction.ToString)
    thelistbox.Items.Add("Solution String =: " + aSolution.solString)
    thelistbox.Items.Add(lastString)
    ListBox1.Items.Add("")
End Sub
Private Sub getInitialSolution(ByRef thelistbox As ListBox, ByVal annealing As Boolean)
    currentSolution = New Solution()
    currentSolution.feasible = False
    Do
        For i = 1 To nInputs
            Randomize()
            currentSolution.itsArray(i) = CInt(Int((2 * Rnd())))
        Next i
```

```
        currentSolution.valueFunctions()
        If currentSolution.costFunction <= upperLimit Then
            currentSolution.feasible = True
            printOut(currentSolution, "Initial feasible solution", thelistbox)
        End If
    Loop Until (currentSolution.feasible = True)
    ' generated an initial solution, currently the best.
    newSolution = New Solution(currentSolution)
End Sub
Private Sub solveByFlop(ByVal coeffN As Integer, ByRef thelistbox As ListBox, ByRef theLB As ListBox)
    Dim looptimes As Integer, count As Integer
    ' Now, atempt to optimize
    statusLabel.Text = "Attempting to optimize current solution ..."
    Do
        ' with simple flip, pick any index at random and flip it
        newSolution.feasible = False
        While (Not newSolution.feasible)
```

```
        'Randomize()
        flipOperation(newSolution)
End While


        ' check profitability of solution just found
    If newSolution.profitFunction > currentSolution.profitFunction Then
        currentSolution = New main.Solution(newSolution)
        printOut(currentSolution, "Feasible improving solution found", thelistbox)
        count = 0
    Else
        printOut(newSolution, "Feasible  Non-improving solution ", thelistbox)
        'newSolution = New Solution(currentSolution)
        count = count + 1
    End If
    looptimes = looptimes + 1
    Label1.Text = "Looped " + looptimes.ToString + "time(s)"
    If count = 100 Then              statusLabel.Text = "FINISHED !  " + Label1.Text
```

```
            ListBox1.Items.Add("*********OPTIMAL SOLUTION**********")
            printOut(currentSolution, "This is the best solution obtained", thelistbox)
        For i = 1 To nInputs
            If currentSolution.itsArray(i) = 1 Then
                theLB.Items.Add(names(i))
              End If
            Next
        End If
    Loop Until count = 100
End Sub

Public Sub generateRandom()
    Dim totalCost = 0
    For i = 1 To nInputs
        'get the name of i
        names(i) = "Item " + i.ToString
        'get the cost of i
```

```
            Randomize()
            cost(i) = CInt(Int(10 + (40 * Rnd())))
            totalCost = totalCost + cost(i)
            'get the profit value of i
            Randomize()
            value(i) = CInt(Int(20 + (50 * Rnd())))
        Next
        numberUpDown.Maximum = nInputs
        upperLimit = CInt(Int((0.4 + (0.3 * Rnd())) * totalCost))
    End Sub
Private Sub nextButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        mainTabControl.SelectTab(mainTabControl.SelectedIndex + 1)
    End Sub
    'end of overviewTabPage subs


    'inputTabPage subs
```

```
    Private Sub acceptTotalButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
acceptTotalButton.Click
        If IsNumeric(totalItemsTextBox.Text) And IsNumeric(totalItemsTextBox.Text) < 1000 Then
            nInputs = Integer.Parse(totalItemsTextBox.Text)
            acceptTotalButton.Enabled = False : totalItemsTextBox.Enabled = False
            inputGroupBox.Enabled = True : clearButton.Enabled = True
            editTotalButton.Enabled = True : submitButton.Enabled = True
            numberTextBox.Text = "1" : AcceptButton = OKButton
            nameTextBox.Focus()
        Else
            MessageBox.Show("Please enter an integer, not more than 1000", "Numeric Input Required", _
                        MessageBoxButtons.OK, MessageBoxIcon.Error)
            totalItemsTextBox.Focus()
        End If
    End Sub
    Private Sub editTotalButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles editTotalButton.Click
        submitButton.Enabled = False : editTotalButton.Enabled = False
```

```
        inputGroupBox.Enabled = False : clearButton.Enabled = False
        totalItemsTextBox.Enabled = True : acceptTotalButton.Enabled = True
        AcceptButton = acceptTotalButton : acceptTotalButton.Focus()
    End Sub

    Private Sub excludeNamesCheckBox_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
excludeNamesCheckBox.CheckedChanged
        If excludeNamesCheckBox.Checked = True Then
            nameTextBox.Enabled = False
            costTextBox.Focus()
        Else
            nameTextBox.Enabled = True
            nameTextBox.Focus()
        End If
    End Sub

    Private Sub OKButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OKButton.Click
```

```
Try
    If Integer.Parse(costTextBox.Text) < 1 Or Integer.Parse(valueTextBox.Text) < 1 Then
        MessageBox.Show("Please enter valid numbers for the cost and value of this item", _
                "Invalid input specified", MessageBoxButtons.OK, MessageBoxIcon.Error)
    Else
        If nextItem > nInputs Then
            MessageBox.Show("Please complete and submit your entries", "Inputs", _
                MessageBoxButtons.OK, MessageBoxIcon.Information)
            upperLimitTextBox.Focus()
            AcceptButton = submitButton
        Else
            cost(nextItem) = Integer.Parse(costTextBox.Text)
            value(nextItem) = Integer.Parse(valueTextBox.Text)
            If excludeNamesCheckBox.Checked = True Then
                names(nextItem) = "Item " + nextItem.ToString
                costTextBox.Focus()
            Else
```

```
            names(nextItem) = nameTextBox.Text
            nameTextBox.Focus()
        End If
        nextItem = nextItem + 1
        numberTextBox.Text = nextItem
      End If
    End If
  Catch ex As Exception
    MessageBox.Show("Please enter valid numbers for the cost and value of this item", _
                    "Invalid input specified", MessageBoxButtons.OK, MessageBoxIcon.Error)
  Finally
    nameTextBox.Clear()
    costTextBox.Clear()
    valueTextBox.Clear()
  End Try
End Sub
```

```
Private Sub clearButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles clearButton.Click
    If MessageBox.Show("This will delete all entries inclding the number of entries involved. " _
            & "Do you still want to clear ALL entries?", "Confirm Delete", _
            MessageBoxButtons.OKCancel, MessageBoxIcon.Warning) = Windows.Forms.DialogResult.OK Then

        totalItemsTextBox.Clear() : excludeNamesCheckBox.Checked = False
        numberTextBox.Clear() : nameTextBox.Clear() : costTextBox.Clear()
        valueTextBox.Clear() : upperLimitTextBox.Clear()
        acceptTotalButton.Enabled = True
    End If
End Sub

Private Sub submitButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles submitButton.Click
    If manualRB.Checked = True Then
        If IsNumeric(upperLimitTextBox.Text) = False Then
            MessageBox.Show("This value must be a number", "Invalid Input", _
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
```

```
        upperLimitTextBox.Focus()
    Else
        upperLimit = Integer.Parse(upperLimitTextBox.Text)
    End If
End If
If upperLimit > 0 Then
    nameListBox.Items.Clear() : costListBox.Items.Clear()
    valueListBox.Items.Clear()
    For i = 1 To nInputs
        nameListBox.Items.Add(names(i).ToString)
        costListBox.Items.Add(cost(i).ToString)
        valueListBox.Items.Add(value(i).ToString)
    Next
    numberUpDown.Maximum = nInputs
    NumericUpDown1.Maximum = nInputs
    Label22.Text = upperLimit.ToString
    mainTabControl.SelectTab(mainTabControl.SelectedIndex + 1)
```

```
        End If
    End Sub

    Private Sub numberTextBox_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
numberTextBox.TextChanged
        If numberTextBox.Text > nInputs Then
            costTextBox.Enabled = False
            valueTextBox.Enabled = False
            upperLimitTextBox.Focus()
            OKButton.Enabled = False
            AcceptButton = submitButton
        End If
    End Sub

    Private Sub manualRB_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
manualRB.CheckedChanged
        If manualRB.Checked = True Then
```

```
        AcceptButton = acceptTotalButton
        manualRB.Enabled = True
      End If
    End Sub

    Private Sub fileRB_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
fileRB.CheckedChanged
      If fileRB.Checked = True Then
        Dim openInput As New OpenFileDialog
        openInput.Filter = "All supported file formats(*.rod;*.xls)|*.rod;*.xls|Resource Optimizer Data File(*.rod)|*.rod|Excel
Worksheet(*.xls)|*.xls"
        openInput.InitialDirectory = "c:\"
        openInput.FilterIndex = 1
        If openInput.ShowDialog = Windows.Forms.DialogResult.OK Then
          Try
            Dim objReader As New StreamReader(openInput.FileName)
            Dim sLine As String = ""
```

```
Dim cnt As Integer = 0, cnt1 As Integer = 1
Do
   sLine = objReader.ReadLine()
   If sLine IsNot Nothing Then
      cnt = cnt + 1
      If cnt < 2 Then
         nInputs = Integer.Parse(sLine.ToString)
      ElseIf cnt < 3 Then
         upperLimit = Integer.Parse(sLine.ToString)
      Else
         If cnt Mod 3 = 0 Then
            names(cnt1) = sLine
         ElseIf cnt Mod 3 = 1 Then
            cost(cnt1) = Integer.Parse(sLine)
         Else
            value(cnt1) = Integer.Parse(sLine)
            cnt1 = cnt1 + 1
```

```
            End If
        End If
    End If
Loop Until sLine Is Nothing
objReader.Close()
MessageBox.Show("The file was loaded successfully. Please submit the data now", "Load Input File successful", _
            MessageBoxButtons.OK, MessageBoxIcon.Information)
If upperLimit > 0 Then
    nameListBox.Items.Clear() : costListBox.Items.Clear()
    valueListBox.Items.Clear()
    For i = 1 To nInputs
        nameListBox.Items.Add(names(i).ToString)
        costListBox.Items.Add(cost(i).ToString)
        valueListBox.Items.Add(value(i).ToString)
    Next
    numberUpDown.Maximum = nInputs
    NumericUpDown1.Maximum = nInputs
```

```
                Label22.Text = upperLimit.ToString
                mainTabControl.SelectTab(mainTabControl.SelectedIndex + 1)
            End If
        Catch ex As Exception
            MessageBox.Show("An error occured. Please wait a short while and try again", "Error reading file", _
                            MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End If
    manualRB.Checked = True
  End If
End Sub

Private Sub randomRB_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
randomRB.CheckedChanged
  If randomRB.Checked = True Then
    Dim nstring As String, result As DialogResult
    nstring = InputBox("How many items / options are to be used?", _
```

```
                    "Enter the value of n to be used", , , )
If nstring = "" Then
    result = Windows.Forms.DialogResult.Cancel
ElseIf Not IsNumeric(nstring) Then
    result = MessageBox.Show("Please enter a valid integer value for n", "Invalid Input", _
                MessageBoxButtons.RetryCancel, MessageBoxIcon.Error)
Else
    nInputs = Integer.Parse(nstring)
    generateRandom()
    If upperLimit > 0 Then
        nameListBox.Items.Clear() : costListBox.Items.Clear()
        valueListBox.Items.Clear()
        For i = 1 To nInputs
            nameListBox.Items.Add(names(i).ToString)
            costListBox.Items.Add(cost(i).ToString)
            valueListBox.Items.Add(value(i).ToString)
        Next
```

```
            numberUpDown.Maximum = nInputs
            NumericUpDown1.Maximum = nInputs
            Label22.Text = upperLimit.ToString
            mainTabControl.SelectTab(mainTabControl.SelectedIndex + 1)
        End If
    End If
    manualRB.Checked = True
  End If
End Sub

Private Sub editButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles editButton.Click
  editGroupBox.Enabled = True
  editButton.Enabled = False
  FOcomputeButton.Enabled = False
End Sub
```

```
    Private Sub NumericUpDown1_ValueChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
NumericUpDown1.ValueChanged
        nameEditBox.Text = names(Integer.Parse(NumericUpDown1.Value.ToString))
        costEditBox.Text = cost(Integer.Parse(NumericUpDown1.Value.ToString))
        valueEditBox.Text = value(Integer.Parse(NumericUpDown1.Value.ToString))
    End Sub
    Private Sub cButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cButton.Click
        editGroupBox.Enabled = False
        editButton.Enabled = True
        FOcomputeButton.Enabled = True
        'SAcomputeButton.Enabled = True
    End Sub

    Private Sub changeButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles changeButton.Click
        Dim theIndex As Integer = NumericUpDown1.Value - 1
        Try
            For i = 0 To nameListBox.Items.Count - 1
```

```
            If nameTextBox.Text = nameListBox.Items.Item(i) And i <> theIndex Then
                Dim ex As System.Exception


            End If
        Next
        If Integer.Parse(costTextBox.Text) < 1 Or Integer.Parse(valueTextBox.Text) < 1 Then
            MessageBox.Show("Cannot replace the entry. The cost or value is not a valid number", "Invalid input specified", _
                            MessageBoxButtons.OK, MessageBoxIcon.Error)
        Else
            If MessageBox.Show("Relace " + nameListBox.Items.Item(theIndex) + ", Cost : " + costListBox.Items.Item(theIndex) + ",
Profit : " + _
                               valueListBox.Items.Item(theIndex) + " with" + ControlChars.NewLine + nameTextBox.Text + ", Cost : " + _
                               costTextBox.Text + "Profit : " + valueTextBox.Text, "Edit entry", MessageBoxButtons.YesNo,
MessageBoxIcon.Exclamation) _
                               = Windows.Forms.DialogResult.Yes Then
                nameListBox.Items.RemoveAt(theIndex)
                costListBox.Items.RemoveAt(theIndex)
```

```
            valueListBox.Items.RemoveAt(theIndex)
            nameListBox.Items.Insert(theIndex, nameEditBox.Text)
            costListBox.Items.Insert(theIndex, costEditBox.Text)
            valueListBox.Items.Insert(theIndex, valueEditBox.Text)
            names(NumericUpDown1.Value) = nameEditBox.Text
            cost(NumericUpDown1.Value) = Integer.Parse(costEditBox.Text)
            value(NumericUpDown1.Value) = Integer.Parse(valueEditBox.Text)
        End If
        editGroupBox.Enabled = False
        editButton.Enabled = True
        FOcomputeButton.Enabled = True
        'SAcomputeButton.Enabled = True
    End If
Catch ex As Exception
    MessageBox.Show("Cannot replace the entry. Invalid number or clashing entries", "Invalid input specified", _
                    MessageBoxButtons.OK, MessageBoxIcon.Error)
End Try
```

```
    End Sub

    Private Sub numberUpDown_ValueChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
numberUpDown.ValueChanged
        Dim indexd As Integer = Integer.Parse(numberUpDown.Value.ToString)
        nameListBox.SelectedItem = names(indexd)
        costListBox.SelectedIndex = nameListBox.SelectedIndex
        valueListBox.SelectedIndex = nameListBox.SelectedIndex
    End Sub
    Private Sub saveInputButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
saveInputButton.Click
        cButton.PerformClick()
        Dim saveThis As New SaveFileDialog
        saveThis.Filter = "Resource Optimizer Data File|*.rod|Excel|*.xls"
        saveThis.Title = "Save Resource Optimizer Input Data"
        saveThis.InitialDirectory = "c:\"
        If saveThis.ShowDialog() = Windows.Forms.DialogResult.OK Then
```

```
' If the file name is not an empty string open it for saving.
If saveThis.FileName <> "" Then
    Select Case saveThis.FilterIndex
        Case 1
            Dim objStreamWriter = New StreamWriter(saveThis.FileName)
            objStreamWriter.WriteLine(nInputs.ToString)
            objStreamWriter.WriteLine(upperLimit.ToString)
            For i = 1 To nInputs
                objStreamWriter.WriteLine(names(i))
                objStreamWriter.WriteLine(cost(i).ToString)
                objStreamWriter.WriteLine(value(i).ToString)
            Next
            objStreamWriter.Close()
            MessageBox.Show("The file was saved successfully", "Save Complete", _
                        MessageBoxButtons.OK, MessageBoxIcon.Information)
        Case 2
            MessageBox.Show("This file cannot be saved in Excel format yet", "Format unavailable", _
```

```
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
            End Select
        End If
    End If
End Sub
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    mainTabControl.SelectTab(mainTabControl.SelectedIndex + 1)
End Sub
Private Sub FOcomputeButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
FOcomputeButton.Click
    mainTabControl.SelectTab(mainTabControl.SelectedIndex + 1)
End Sub
Private Sub CBox_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CBox.CheckedChanged
    If CBox.Checked = True Then
        yTextBox.ReadOnly = False
    Else
```

```
        yTextBox.ReadOnly = True
    End If
End Sub
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
    ListBox1.Items.Clear()
    lb2.Items.Clear()
    getInitialSolution(ListBox1, False)
    solveByFlop(2, ListBox1, lb2)
End Sub
Private Sub saveResults_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim k As Integer
    Dim s As String = ""
    For k = 0 To ListBox1.Items.Count - 1
        s = s + ListBox1.Items(k).ToString + vbCrLf
    Next
    s = s + "**********Selected Items*********" + vbCrLf
    For k = 0 To Me.lb2.Items.Count - 1
```

```
            s = s + lb2.Items(k).ToString + vbCrLf
        Next
        Dim SaveFileDialog As New SaveFileDialog
        SaveFileDialog.InitialDirectory = My.Computer.FileSystem.SpecialDirectories.MyDocuments
        SaveFileDialog.Filter = "Text Files (*.txt)|*.txt|All Files (*.*)|*.*"
        If (SaveFileDialog.ShowDialog(Me) = System.Windows.Forms.DialogResult.OK) Then
            Dim FileName As String = SaveFileDialog.FileName
            My.Computer.FileSystem.WriteAllText(FileName, s, True)
            MessageBox.Show("Results saved succesfully", "Save Complete", MessageBoxButtons.OK, MessageBoxIcon.Information)
        End If
    End Sub


End Class
```