

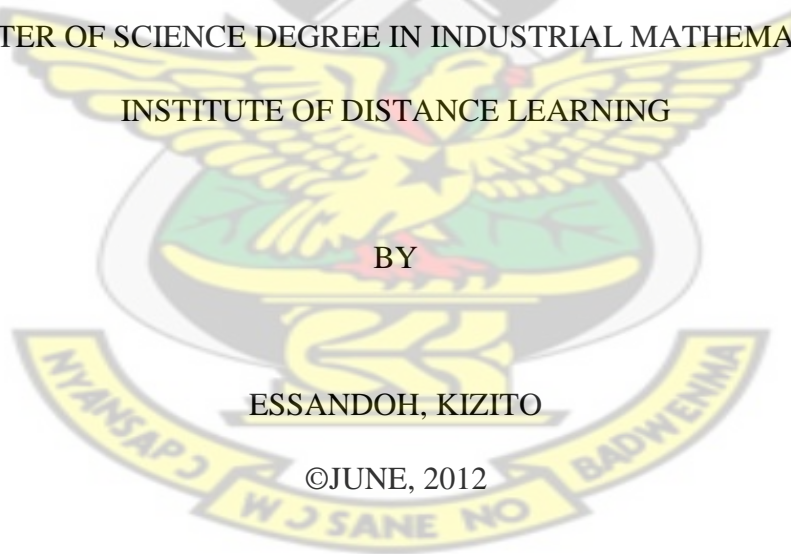
***MODELING SITE DEVELOPMENT FOR GARBAGE DISPOSAL AS A 0-1
KNAPSACK PROBLEM, A CASE STUDY OF THE SEKONDI-TAKORADI METROPOLIS***

A THESIS SUBMITTED TO THE DEPARTMENT OF MATHEMATICS,
KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY
KUMASI, GHANA
IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD
OF
MASTER OF SCIENCE DEGREE IN INDUSTRIAL MATHEMATICS
INSTITUTE OF DISTANCE LEARNING

BY

ESSANDOH, KIZITO

©JUNE, 2012



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.

KNUST

Kizito Essandoh PG4065410

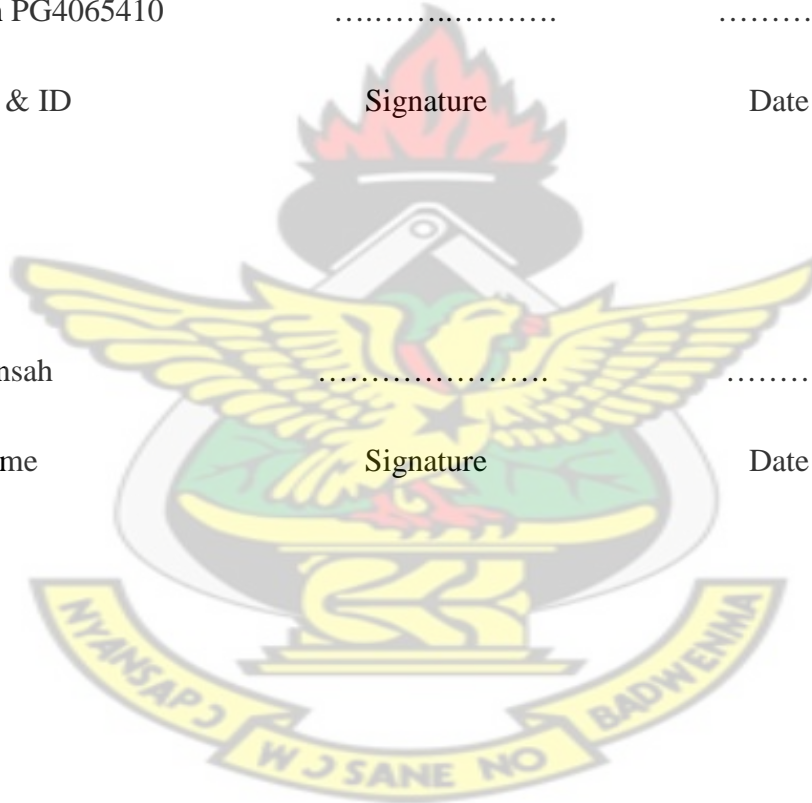
Student's Name & ID Signature Date

Certified By
Dr. S. K. Amponsah

Supervisor's Name Signature Date

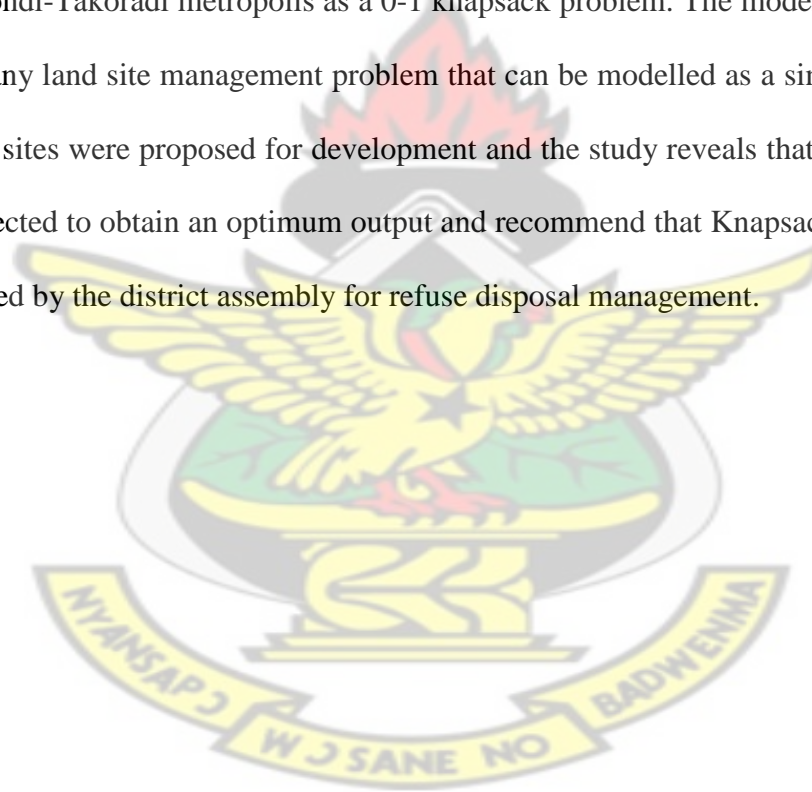
Certified By
Mr. Kwaku Darkwa

Head of Department's Name Signature Date



ABSTRACT

Integer programming is an important class of mathematical programming problems which is a useful tool for modelling and optimizing real-life problems. The knapsack problem is a form of integer programming problem that has only one constraint and can be used to strengthen cutting planes for general integer programs. These facts make the studies of the knapsack problems and their variants extremely important area of research in the field of operations research. This thesis seeks to apply the branch-and-bound algorithm to model site development for solid waste disposal in Sekondi-Takoradi metropolis as a 0-1 knapsack problem. The model developed could be adopted for any land site management problem that can be modelled as a single 0-1 knapsack problem. Seven sites were proposed for development and the study reveals that sites A, B, F and G should be selected to obtain an optimum output and recommend that Knapsack problem model should be adopted by the district assembly for refuse disposal management.



DEDICATION

To the Glory of Almighty God, I dedicate this project to my dear wife Jemima Boyan and my lovely son C.K. Essandoh. Also to all my colleagues especially Franscio Mohammed

KNUST



ACKNOWLEDGEMENT

I would like to give thanks to the Almighty God for granting me the strength and knowledge to complete this course successfully. I am very grateful to my supervisor, Dr. S.K. Amponsah of the Department of Mathematics, KNUST, who was always ready to assist me in one way or the other and also helped me in the compilation of this work, may God richly bless him. I also wish to express my profound gratitude to all the lecturers at the mathematics Department who contributed in one way or the other for the successful completion of this project. I also give thanks to my wife Jemima Boyan for her support during the entire course.

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

God richly bless you all.

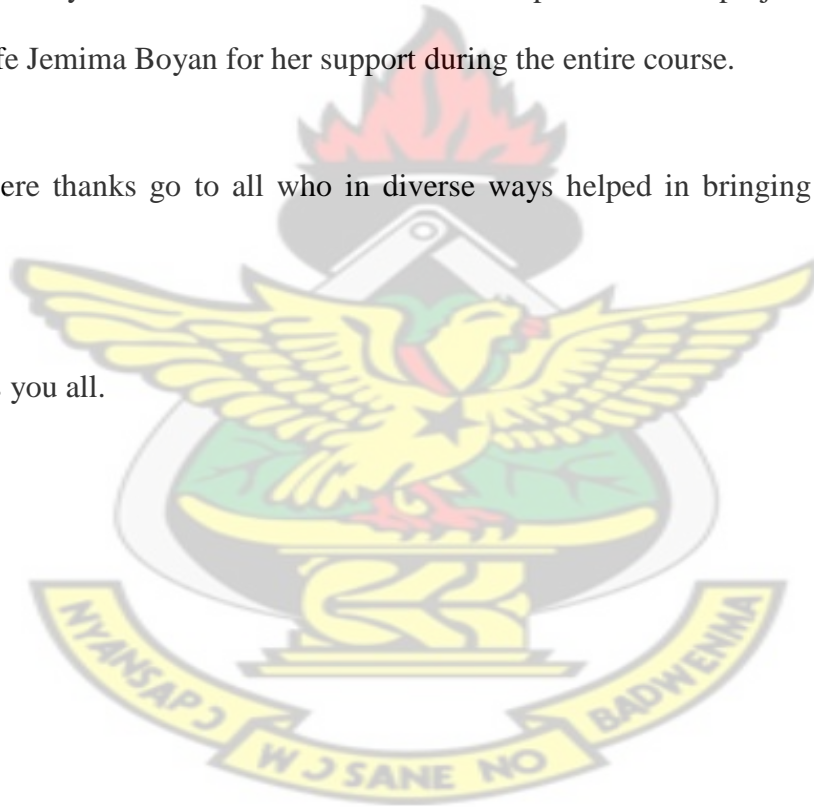


TABLE OF CONTENTS

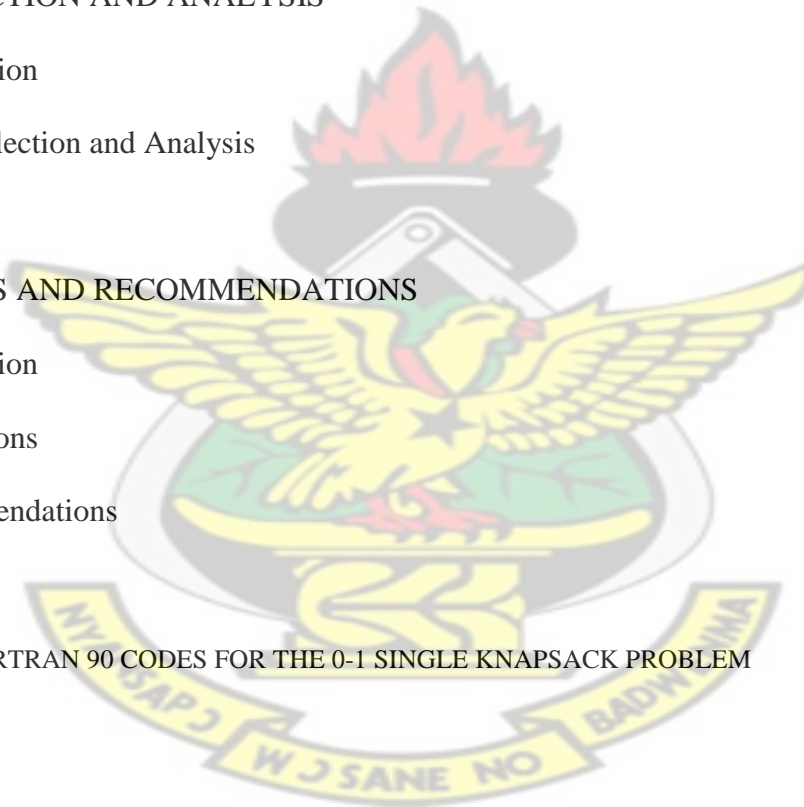
DECLARATION	i
ABSTRACT	ii
DEDICATION	iii
ACKNOWLEDGEMENT	iv
TABLE OF CONTENTS	v
LIST OF TABLES	vi
CHAPTER 1	1
INTRODUCTION	1
1.1 Background of Study	1
1.2 Problem Statement	4
1.3 Objectives	5
1.4 Methodology	6
1.5 Justification	6
1.6 limitations of the study	6
1.7 organization of the study	7
CHAPTER 2	8
LITERATURE REVIEW	8

KNUST



CHAPTER 3	31
METHODOLOGY	31
3.0 Introduction	31
3.1 Branch-and-Bound Algorithm for Knapsacks	33
3.2 The Horowitz-Sanni Algorithm	34
3.3 The Martellow-Toth Algorithm	37
CHAPTER 4	43
DATA COLLECTION AND ANALYSIS	43
4.0 Introduction	43
4.1 Data Collection and Analysis	44
CHAPTER 5	47
CONCLUSIONS AND RECOMMENDATIONS	47
5.0 Introduction	47
5.1 Conclusions	47
5.2 Recommendations	47
REFERENCES	49
APPENDIX_1-FORTRAN 90 CODES FOR THE 0-1 SINGLE KNAPSACK PROBLEM	55

KNUST



LIST OF TABLE

4.1 List of the capacity (tons/week) and the cost (1000/unit) for each site	44
4.2 Optimal solutions for the various iterative stages	46

KNUST



CHAPTER 1

INTRODUCTION

Knapsack problems are among the most intensively studied NP-hard combinatorial optimization problems. The applications of these problems span a wide canvas from industrial applications and financial management to electronic commerce and personal health-care. The common flavour in most of these problems is resource allocation. The allocation of a specific amount of a single resource among competitive alternatives is often modelled as a knapsack problem or its variants. In this chapter of the thesis, we shall give an overview of integer programming model of which Knapsack problems falls under; a brief description of the problem statement of the thesis is also presented as well as the objectives, the methodology, the justification and the organization of the thesis.

1.1 BACKGROUND OF STUDY

Mathematical programming is a fast growing branch of mathematics with a surprisingly short history. Most of its development has occurred during the second half of this century. Basically, one deals with the maximization (or minimization) of some function subject to one or more constraints.

Today mathematical programming problems arise in all sorts of areas; this is the age of optimization as a scientist stated (Geir, 1997). Modern society, with advanced technology and competitive businesses typically needs to make best possible decisions, which e.g. involve the best possible use of resources, maximizing some revenue, minimizing production or design costs, etc. In mathematical areas one may meet approximation problems like solving some equations “within some tolerance” but without using too many variables (resources). In computer

science, the very large scale integration (VLSI) area gives rise to many optimization problems: physical layout of microchips, routing, via minimization and so on. In telecommunications, the physical design of networks lead to many different optimization problems, e.g. that of minimizing network design (or expansion) costs subject to constraints reflecting that the network can support the described traffic. In fact, in many other areas, problems involving communication networks can be viewed as optimization problems. In economics (econometrics) optimization models are used for e.g. describing money transfer between sectors in society or describing the efficiency of production units.

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

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

With the standard linear programming problem, the assumption that choice variables are infinitely divisible (can be any real number) is unrealistic in many settings. Integer programming problems are typically much harder to solve than linear programming problems and there are no fundamental theoretical results like Duality or Computational algorithms like the Simplex algorithm to help one to understand and solve the problems. This sad realization has made the study of integer programming problems goes in two directions. First, people study specialized model. These problems can be solved as linear programming problems (that is, adding the

integer constraints does not change the solution). In many cases, they can be solved more efficiently than general linear programming problems using new algorithms. Second, people introduce general algorithms. These algorithms are not as computationally efficient as the simplex algorithm, but can be formulated generally.

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

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

The computerized system that was developed used a mathematical model to incorporate each of the goals and increased San Francisco Police Department's net income by 14 million dollars and decreased response times by twenty (20) percent. Similarly, Delta Airlines saved approximately 100 million dollars per year by implementing optimal fleet assignments. More than 2,500 domestic flights and 450 airplanes per day are assigned by this integer programming (IP) (Scheff et al., 1994).

In addition to the above application, integer programs have been used to solve a number of real-life problems, including airline scheduling (Gutierrez, 2007), and (Huschka, 2007), sports scheduling (Easton, et al., 2003), construction site location (Nemhauser and Wolsey, 1988), manufacturing job scheduling, and telephone network optimizations (Tomastik, 1993). Thus,

integer programming has played an important role in supporting managerial decisions in the areas of capital budgeting, warehouse location, and scheduling.

The Knapsack Problems are among the simplest integer programming problems which are NP-hard. Problems in this class are typically concerned with selecting from a set of given items, each with a specific weight and value, a subset of items whose weight sum does not exceed a prescribed capacity and whose value is maximum. The specific problem that arises depends on the number of knapsack (single or multiple) to be filled and on the number of available items of each type (bounded or unbounded). Because of their wide range of applicability, knapsack problems have known a large number of variations such as: single and multiple constrained knapsacks, knapsack with disjunctive constraints, multidimensional knapsacks, multiple choice knapsacks, single and multiple objective knapsacks, integer, linear, non-linear knapsacks, deterministic and stochastic knapsacks, knapsacks with convex / concave objective functions, etc.

The 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-five (45) years because of their direct application to problems arising in industries 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.

1.2 PROBLEM STATEMENT

This thesis seeks to optimally select the best site among the various proposed sites that have been ear-marked for refuse disposal in Sekondi-Takoradi Metropolis given the budget constraint.

An example of this problem is a camper going backpacking. He wishes to bring the best combination of equipment he can. Each piece of equipment (tent, food, water, etc) has a value to the camper that is assigned a numerical representation. Each piece of equipment also has a corresponding weight, but the capacity of the bag is b . The camper can only bring as much equipment as he can carry. There are various items he can carry in the bag, but the total weight of these items is greater than the weight the bag can carry. If each item has a value, v_i and a weight w_i for each i (such that $i = 1, 2, 3, \dots, N$, where N is the total number of items) and x_i the number of units of item i in the bag.

Two examples of areas where knapsack problems can be applied are resource allocation (Granmo et al., 2007) and portfolio management (Bertsimas et al., 1999). In resource allocation, a company wishes to maximize its return from resources invested into each division or product subject to the total resources available. In portfolio management, the goal is to maximize returns while minimizing risk. The knapsack problem is widely studied because of its importance to integer programs. Any single constraint of a binary integer program can be viewed as a knapsack constraint.

1.3 OBJECTIVES

The objective of the study is to model a real-life problem in site development for rubbish disposal as a 0-1 knapsack problem, and propose branch-and-bound algorithm for solving the problem.

1.4 METHODOLOGY

This thesis seeks to apply the branch-and-bound algorithm for solving our proposed knapsack problem. First, the algorithm is presented along with relevant examples. A real life computational study is performed and a code in FORTRAN 90 programming language will be employed to implement the algorithm.

1.5 JUSTIFICATION

Knapsack problems are widely used in financial decision making, and very interesting from the perspective of computer science because; there is a pseudo polynomial time algorithm using dynamic programming, there is a fully polynomial-time approximation scheme, which uses the pseudo-polynomial time algorithm as a subroutine, the problem is NP-complete to solve exactly, thus it is expected that no known algorithm can be both correct and fast (polynomial-time) on all cases, and many cases that arise in practice, and “random instances” from some distributions, can nonetheless be solved exactly. In view of these, studies of knapsack problems and their algorithms has been an area of much interest in the contribution to academic knowledge, hence the reason for solving the knapsack problem.

1.6 LIMITATIONS OF THE STUDY

The revenue generation or the income of the assembly is most at times uncertain and can affect the budgeted amount for refuse management. Also, the rate of inflation is unpredictable which could lead to high cost of goods and services.

1.7 ORGANIZATION OF THE THESIS

The study is organized in five chapters. In chapter one, we presented a background study of integer programming of which knapsack problems form part, the problem statement, the objectives, methodology, justification and limitation of the study.

In chapter two, pertinent literature in the field Knapsack problems will be discussed.

Chapter three presented the branch-and-bound algorithm.

Chapter four is data collection and analysis

Chapter five, the last chapter presents conclusion and recommendations of the study.



CHAPTER 2

LITERATURE REVIEW

The knapsack problem is a classical combinatorial problem used to model many industrial situations. Faced with uncertainty on the model parameters, robustness analysis is an appropriate approach to find reliable solutions. Kalai and Vanderpooten (2006) studied the robust knapsack problem using a max-min criterion, and proposed a new robustness approach, called lexicographic α -robustness. The authors showed that the complexity of the lexicographic α -robust problem does not increase compared with the max-min version and presented a pseudo-polynomial algorithm in the case of a bounded number of scenarios.

Knapsack problems with setups find their application in many concrete industrial and financial problems. Moreover, they also arise as sub-problems in a Dantzig-Wolfe decomposition approach to more complex combinatorial optimization problems, where they need to be solved repeatedly and therefore efficiently. Micheal et al., (2009) considered the multiple-class integer knapsack problem with setups. Items are partitioned into classes whose use imply a setup cost and associated capacity consumption. Item weights are assumed to be a multiple of their class weight. The total weight of selected items and setups is bounded. The objective is to maximize the difference between the profits of selected items and the fixed costs incurred for setting-up classes. A special case is the bounded integer knapsack problem with setups where each class holds a single item and its continuous version where a fraction of an item can be selected while incurring a full setup. The authors showed the extent to which classical results for the knapsack problem can be generalized to these variants with setups. In particular, an extension of the

branch-and-bound algorithm of Horowitz and Sahni (1974) is developed for problems with positive setup costs.

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

The Quadratic Knapsack Problem (QKP) calls for maximizing a quadratic objective function subject to a knapsack constraint, where all coefficients are assumed to be nonnegative and all variables are binary. The problem has applications in location and hydrology, and generalizes the problem of checking whether a graph contains a clique of a given size. Alberto et al., (2007) proposed an exact branch-and-bound algorithm for QKP, where upper bounds are computed by considering a Lagrangian relaxation that is solvable through a number of (continuous) knapsack problems. Suboptimal Lagrangian multipliers are derived by using sub-gradient optimization and provide a convenient reformulation of the problem. The authors also discussed the relationship between our relaxation and other relaxations. Heuristics, reductions, and branching schemes were described. In particular, the processing of each node of the branching tree is quite fast: Their approach does not update the Lagrangian multipliers, and use suitable data structures to

compute an upper bound in linear expected time in the number of variables. The authors reported exact solution of instances with up to 400 binary variables, i.e., significantly larger than those solvable by the previous approaches. The key point of this improvement is that the upper bounds we obtain are typically within 1% of the optimum, but can still be derived effectively. They also showed that their algorithm is capable of solving reasonable-size Max Clique instances.

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

Opong (2009) presented 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. The author approach is flexible and can incorporate the use of the knapsack for Profit maximization in the TV adverts selection problem, and focused on using a simple heuristic scheme (Simple flip) for the solution of knapsack problems.

The collapsing knapsack problem is a generalization of the ordinary knapsack problem, where the knapsack capacity is a non-increasing function of the number of items included. Whereas previous methods on the topic have applied quite involved techniques,

Ulrich et al., (1995) presented and analyze two rather simple approaches: One approach that was based on the reduction to a standard knapsack problem, and another approach that was based on a simple dynamic programming recursion. Both algorithms have pseudo-polynomial solution times, guaranteeing reasonable solution times for moderate coefficient sizes. Computational experiments are provided to expose the efficiency of the two approaches compared to previous algorithms.

Kosuch and Lisser (2009) studied a particular version of the stochastic knapsack problem with normally distributed weights: the two-stage stochastic knapsack problem. Contrary to the single-stage knapsack problem, items can be added to or removed from the knapsack at the moment the actual weights become known (second stage). In addition, a chance-constraint is introduced in the first stage in order to restrict the percentage of cases where the items chosen lead to an overload in the second stage. According to the authors, there is no method known to exactly

evaluate the objective function for a given first-stage solution, and therefore proposed methods to calculate the upper and lower bounds. These bounds are used in a branch-and-bound framework in order to search the first-stage solution space. Special interest was given to the case where the items have similar weight means. Numerical results are presented and analyzed.

Stefanie (2010) presented an Ant Colony Optimization algorithm for the Two-Stage Knapsack problem with discretely distributed weights and capacity, using a meta-heuristic approach. Two heuristic utility measures were proposed and compared. Moreover, the author introduced the novel idea of non-utility measures in order to obtain a criterion for the construction termination. The author argued why for the proposed measures, it is more efficient to place pheromone on arcs instead of vertices or edges of the complete search graph. Numerical tests show that the author's algorithm is able to produce, in much shorter computing time, solutions of similar quality than CPLEX after two hour. Moreover, with increasing number of scenarios the percentage of runs where his algorithm is able to produce better solutions than CPLEX (after 2h) increases.

Mattfeld and Kopfer (2003) described terminal operations for the vehicle transshipment hub in Bremerhaven as a knapsack and have derived an integral decision model for manpower planning and inventory control. The authors proposed a hierarchical separation of the integral model into sub models and can develop integer programming algorithm to solve the arising sub problems.

In bus transit operations planning process, the important components are network route design, setting timetables, scheduling vehicles, assignment of drivers, and maintenance scheduling.

Haghani and Shafahi (2002) presented integer programming model to design daily inspection and maintenance schedules for the buses that are due for inspection so as to minimize the

interruptions in the daily bus operating schedule, and maximize the utilization of the maintenance facilities.

The setting of timetables and bus routing or scheduling are essential to an intercity bus carrier's profitability, its level of service, and its competitive capacity in the market. Yan and Chen (2002) developed a model that help Taiwanese intercity bus carriers in timetable settings and bus routing or scheduling. The model employs multiple time-space networks that can formulate bus movements and passenger flows and manage the interrelationships between passenger trip demands and bus trip suppliers to produce the best timetables and bus routes or schedules.

Higgins et al., (1996) described the development and use of integer programming model to optimize train schedules on single-line rail corridors. The model has been developed with two major applications in mind: as a decision support tool for train dispatchers to schedule trains in real time in an optimal way and as a planning tool to evaluate the impact of timetable changes, as well as railroad infrastructure changes. The model was developed based on a real-life problem.

Ghoseiri et al., (2004) developed an optimization model for the passenger train-scheduling problem on a railroad network, which includes single, and multiple tracks, as well as multiple platforms with defferent train capacities.

Claessens et al., (1998) considered the problem of cost optimal railway line allocation for passenger trains for the Dutch railway system. A mathematical programming model was developed, which minimized the operating costs subject to service constraints and capacity requirements. The model optimized on lines, line types, routes, frequencies, and train lengths. First, the line allocation model was formulated as an integer nonlinear programming model. The

model was then transformed into an integer linear programming model with binary decision variables. The model was solved and applied to a sub network of the Dutch railway system for which it showed a substantial cost reduction.

The deterministic knapsack problem is a well known and well studied NP-hard combinatorial optimization problem. It consists in filling a knapsack with items out of a given set such that the weight capacity of the knapsack is respected and the total reward maximized. In the deterministic problem, all parameters (item weights, rewards, knapsack capacity) are known (deterministic). In the stochastic counterpart, some (or all) of these parameters are assumed to be random, i.e. not known at the moment the decision has to be made.

Stefanie et al., (2010) studied the stochastic knapsack problem with expectation constraint. The item weights are assumed to be independently normally distributed. The authors solved the relaxed version of this problem using a stochastic gradient algorithm in order to provide upper bounds for a branch-and-bound framework. Two approaches to estimate the needed gradients are applied, one based on Integration by Parts and one using Finite Differences. Finite Differences is a robust and simple approach with efficient results despite the fact that the estimated gradients are biased; meanwhile Integration by Parts is based upon a more theoretical analysis and permits to enlarge the field of applications.

Stefanie et al., (2009) proposed a mixed integer bi-level problem having a probabilistic knapsack constraint in the first level. The problem formulation is mainly motivated by practical pricing and service provision problems as it can be interpreted as a model for the interaction between a service provider and clients. The authors assumed the probability space to be discrete which allows us to reformulate the problem as a deterministic equivalent bi-level problem. Via a

reformulation as linear bi-level problem, we obtain a quadratic optimization problem, the so called Global Linear Complementarity Problem. Based on this quadratic problem, the authors finally proposed a procedure to compute upper bounds on the initial problem by using a Lagrangian relaxation and an iterative linear min-max scheme.

The knapsack problem (KP) and its multidimensional version (MKP) are basic problems in combinatorial optimization.

KNUST

Thibaut and Jacques (2010) presented the multiobjective extension (MOKP and MOMKP), for which the aim is to obtain or to approximate the set of efficient solutions. In a first step, the authors classified and described briefly the existing works that are essentially based on the use of meta-heuristics. In a second step, the authors proposed the adaptation of the two-phase Pareto local search (2PPLS) to the resolution of the MOMKP. With this aim, the authors used a very-large scale neighborhood (VLSN) in the second phase of the method that is the Pareto local search. They compared their results to state-of-the-art results and showed that they obtained results never reached before by heuristics, for the biobjective instances. Finally they considered the extension to three-objective instances.

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

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

Lawler (1997) presented fully polynomial approximation algorithms for knapsack problems are presented. These algorithms are based on ideas of Ibarra and Kim, with modifications which yield better time and space bounds, and also tend to improve the practicality of the procedures. Among the principal improvements are the introduction of a more efficient method of scaling and the use of a median-finding routine to eliminate sorting. The 0-1 knapsack problem, for n items and accuracy $\varepsilon > 0$, is solved in $(n \log (1/\varepsilon) + 1/\varepsilon^4)$ time and $O(n + 1/\varepsilon^3)$ space. The time bound is reduced to $O(n + 1/\varepsilon^3)$ for the "unbounded" knapsack problem. For the "subset-sum" problem, $O(n + 1/\varepsilon^3)$ times and $O(n + 1/\varepsilon^2)$ spaces, or $O(n + 1/\varepsilon^2 \log (1/\varepsilon))$ time and space, are achieved. The "multiple choice" problem, with m equivalence classes, is solved in $O(nm^2/\varepsilon)$ time and space.

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

Esther et al., (1993) studied a variety of geometric versions of the classical knapsack problem.

In particular, the authors considered the following “fence enclosure” problem: given a set S of n points in the plane with values $v_i > 0$, we wish to enclose a subset of the points with a fence (a simple closed curve) in order to maximize the “value” of the enclosure. The value of the enclosure is defined to be the sum of the values of the enclosed points minus the cost of the fence. They also considered various versions of the problem, such as allowing S to consist of points and/or simple polygons. Other versions of the problems are obtained by restricting the total amount of fence available and also allowing the enclosure to consist of at most M connected components. When there is an upper bound on the length of fence available, we show that the problem is NP-complete. We also provide polynomial-time algorithms for many versions of the fence problem when an unrestricted amount of fence is available.

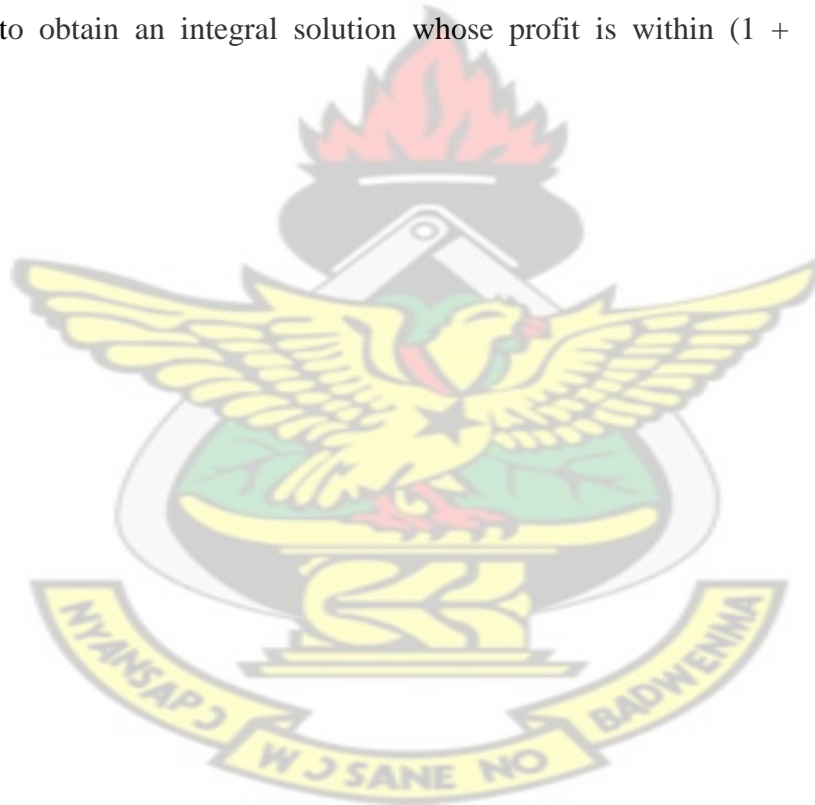
Volgenant and Zoon (1990) presented a multidimensional 0-1 knapsack problem using heuristic, based on Lagrange multipliers, that also enables the determination of an upper bound to the optimal criterion value. This heuristic is extended in two ways: (1) in each step, not one, but more multiplier values are computed simultaneously, and (2) at the end the upper bound is sharpened by changing some multiplier values. From a comparison using a large series of different test problems, the extensions appear to yield an improvement, on average, at the cost of only a modest amount of extra computing time.

The binary knapsack problem is a combinatorial optimization problem in which a subset of a given set of elements needs to be chosen in order to maximize profit, given a budget constraint.

Das and Ghosh (2003) studied a stochastic version of the problem in which the budget is random. The authors proposed two different formulations of this problem, based on different ways of handling infeasibility, and propose an exact algorithm and a local search-based heuristic to solve the problems represented by these formulations. The authors also presented

the results from some computational experiments.

Goyal and Ravi (2009) presented a stochastic knapsack problem where each item has a known profit but a random size. The goal is to select a profit maximizing set of items such that the probability of the total size of selected items exceeding the knapsack size is at most a given threshold. The authors presented a parametric linear programming (LP) formulation and showed that it is a good approximation of the chance-constrained stochastic knapsack problem. Furthermore, they gave a polynomial time algorithm to round any fractional solution of the parametric LP to obtain an integral solution whose profit is within $(1 +$



The knapsack problem is known to be a typical NP-complete problem, which has 2^n possible solutions to search over. Thus a task for solving the knapsack problem can be accomplished in 2^n trials if an exhaustive search is applied. In the past decade, much effort has been devoted in order to reduce the computation time of this problem instead of exhaustive search. In 1984, Karnin proposed a brilliant parallel algorithm, which needs $O(2^{n/6})$ processors to solve the knapsack problem in $O(2^{n/2})$ time; that is, the cost of Karnin's parallel algorithm is $O(2^{2n/3})$. Der-Chyuan Lou and Chin-Chen Chang (1997) proposed a fast search technique to improve Karnin's parallel algorithm by reducing the search time complexity of Karnin's parallel algorithm to be $O(2^{n/3})$ under the same $O(2^{n/6})$ processors available. Thus, the cost of the proposed parallel algorithm is $O(2^{n/2})$. Furthermore, the authors extended their technique to the case that the number of available processors is $P = O(2^x)$, where $x \geq 1$. From the analytical results, they saw that their search technique is indeed superior to the previously proposed methods. They do believe their proposed parallel algorithm is pragmatically feasible at the moment when multiprocessor systems become more and more popular.

Knapsack problem is a typical NP complete problem. During last few decades, Knapsack problem has been studied through different approaches, according to the theoretical development of combinatorial optimization. Garg and Sunanda (2009) put forward the evolutionary algorithm for 0/1 knapsack problem. A new objective function evaluation operator was proposed which employed adaptive repair function named as repair and elitism operator to achieve optimal results in place of problem specific knowledge or domain specific operator like penalty operator (which are still being used). Additional features had also been incorporated which allowed the algorithm to perform more consistently on a larger set of problem instances.

Their study also focused on the change in behavior of outputs generated on varying the crossover and mutation rates. New algorithm exhibited a significant reduction in number of function evaluations required for problems investigated.

Srisuwannapa and Charnsethikul (2007) presented a variant of the unbounded knapsack problem (UKP) into which the processing time of each item is also put and considered, referred as MMPTUKP. The MMPTUKP is a decision problem of allocating amount of n items, such that the maximum processing time of the selected items is minimized and the total profit is gained as at least as determined without exceeding capacity of knapsack. In this study, we proposed a new exact algorithm for this problem, called MMPTUKP algorithm. This pseudo polynomial time algorithm solves the bounded knapsack problem (BKP) sequentially with the updated bounds until reaching an optimal solution. The authors presented computational experience with various data instances randomly generated to validate their ideas and demonstrate the efficiency of the proposed algorithm.

Ronghua et al., (2006) presented a new multiobjective optimization (MO) algorithm to solve 0/1 knapsack problems using the immune Clonal principle. This algorithm is termed Immune Clonal MO Algorithm (ICMOA). In ICMOA, the antibody population is split into the population of the non-dominated antibodies and that of the dominated anti-bodies. Meanwhile, the non-dominated antibodies are allowed to survive and to clone. A metric of Coverage of Two Sets are adopted for the problems. This quantitative metric is used for testing the convergence to the Pareto-optimal front. Simulation results on the 0/1 knapsack problems show that ICMOA, in most problems, is able to find much better spread of solutions and better convergence near the true Pareto-optimal front compared with SPEA, NSGA, NPGA and VEGA.

Deniz et al., (2010) studied maximization of revenue in the dynamic and stochastic knapsack problem where a given capacity needs to be allocated by a given deadline to sequentially arriving agents. Each agent is described by a two-dimensional type that reflects his capacity requirement and his willingness to pay per unit of capacity. Types are private information. The authors first characterize implementable policies. Then they solved the revenue maximization problem for the special case where there is private information about per-unit values, but capacity needs are observable. After that they derived two sets of additional conditions on the joint distribution of values and weights under which the revenue maximizing policy for the case with observable weights is implementable, and thus optimal also for the case with two-dimensional private information. In particular, they investigated the role of concave continuation revenues for implementation. We also construct a simple policy for which per-unit prices vary with requested weight but not with time, and prove that it is asymptotically revenue maximizing when available capacity/ time to the deadline both go to infinity. This highlights the importance of nonlinear as opposed to dynamic pricing.

Computational grids are distributed systems composed of heterogeneous computing resources which are distributed geographically and administratively. These highly scalable systems are designed to meet the large computational demands of many users from scientific and business orientations. However, there are problems related to the allocation of the computing resources which compose of a grid.

Van dester et al., (2008) studied the design of a Pan-Canadian grid. The design exploits the maturing stability of grid deployment toolkits, and introduces novel services for efficiently allocating the grid resources. The changes faced by this grid deployment motivate further exploration in optimizing grid resource allocations. By applying this model to the grid

allocation option, it is possible to quantify the relative merits of the various possible scheduling decisions. Using this model, the allocation problem was formulated as a knapsack problem. Formulation in this manner allows for rapid solution times and results in nearly optimal allocations.

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

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

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

discrete-dynamic model for handling single-night reservation requests, based on a static integer programming model, developed to handle multi-night reservation requests.

Allocation of resources under uncertainty is a very common problem in many real-life scenarios. Employers have to decide whether or not to hire candidates, not knowing whether future candidates will be stronger or more desirable. Machines need to decide whether to accept jobs without knowledge of the importance or profitability of future jobs. Consulting companies must decide which jobs to take on, not knowing the revenue and resources associated with potential future requests. More recently, online auctions have proved to be a very important resource allocation problem. Advertising auctions in particular provide the main source of monetization for a variety of internet services including search engines, blogs, and social networking sites. Additionally, they are the main source of customer acquisition for a wide array of small online business, of the networked world. In bidding for the right to appear on a web page (such as a search engine), advertisers have to trade off between large numbers of parameters, including keywords and viewer attributes. In this scenario, an advertiser may be able to estimate accurately the bid required to win a particular auction, and benefit either in direct revenue or name recognition to be gained, but may not know about the trade off for future auctions. All of these problems involve an online scenario, where an algorithm has to make decisions on whether to accept an offer, based solely on the required resource investment (or weight) and projected value of the current offer, with the total weight of all selected offer not exceeding a given budget. When the weights are uniform and equal to the weight constraint, the problems above reduce to the famous secretary problem which was first introduced by (Dynkin, 1963). Moshe et al., (2008), studied this model as a knapsack problem.

Kleinberg (2009) presented a model for the multiple-choice secretary problem in which k elements need to be selected and the goal is to maximize the combined value (sum) of the selected elements.

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

Aggarwal and Hartline (2006) designed truthful auctions which are revenue competitive when the auctioneer is constrained to choose agents with private values and publicly known weights that fit into a knapsack.

Boryczka (2006) presented a new optimization algorithm based on ant colony metaphor and a new approach for the Multiple Knapsack Problem. The MKP is the problem of assigning a subset of n items to m distinct knapsacks, such that the total profit sum of the selected items is maximized, without exceeding the capacity of each of the knapsacks. The problem has several difficulties in adaptation as well as the trail representation of the solutions of MKP or a dynamically changed heuristic function applied in this approach. Presented results showed the power of the ACO approach for solving this type of subset problems.

The Multiple-Choice Multi-Dimension Knapsack Problem (MMKP) is a variant of the 0-1 knapsack problem, an NP-Hard problem. Due to its high computational complexity, algorithms for exact solution of the MMKPs are not suitable for most real-time decision-making applications, such as quality adaptation and admission control for interactive multimedia systems, or service level agreement (SLA) management in telecommunication networks.

Shahadat et al., (2002) presented a heuristic for finding near-optimal solutions of the MMKP, with reduced computational complexity, and is suitable for real-time applications. Based on Toyoda's concept of aggregate resource, the heuristic employs an iterative improvement procedure using savings in aggregate resource and value per unit of extra aggregate resource. Experimental results suggest that this heuristic finds solutions which are close to the optimal (within 6% of the optimal value), and that it out-performs Moser's heuristic for the MMKP in both solution quality and execution time.

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

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

packing heuristics compare favorably with the best reported efforts in the literature. Additionally, the author proposed the JAVA language to be an effective language for implementing the heuristics. The search is then used in a real world problem of determining how much cargo can be packed with a given fleet of aircraft.

Knapsack problem has been widely studied in computer science for years. There exist several variants of the problem, with zero-one maximum knapsack in one dimension being the simplest one.

Islam (2009) studied several existing approximation algorithms for the minimization version of the problem and proposed a scaling based fully polynomial time approximation scheme for the minimum knapsack problem. The author compared the performance of this algorithm with existing algorithms. His experiments show that, the proposed algorithm runs fast and has a good performance ratio in practice. He also conducts extensive experiments on the data provided by Canadian Pacific Logistics Solutions during the MITACS internship program. The author proposed a scaling based varepsilon-approximation scheme for the multidimensional (d -dimensional) minimum knapsack problem and compares its performance with a generalization of a greedy algorithm for minimum knapsack in d -dimensions. The author's experiments showed that the varepsilon-approximation scheme exhibits good performance ratio in practice.

Maya and Dipti (2011) presented a research project on using Genetic Algorithms (GAs) to solve the 0-1 Knapsack Problem (KP). The Knapsack Problem is an example of a combinatorial optimization problem, which seeks to maximize the benefit of objects in a

knapsack without exceeding its capacity. The author's research contains three sections: brief description of the basic idea and elements of the GAs, definition of the Knapsack Problem, and implementation of the 0-1 Knapsack Problem using GAs. The main focus of the research was on the implementation of the algorithm for solving the problem. In the program, he implemented two selection functions, roulette-wheel and group selection. The results from both of them differed depending on whether to use elitism or not. Elitism significantly improved the performance of the roulette-wheel function. Moreover, the author tested the program with different crossover ratios and single and double crossover points but the results given were not that different.

Maya and Dipti (2005) studied several algorithm design paradigms applied to a single problem – the 0/1 Knapsack Problem. The Knapsack problem is a combinatorial optimization problem where one has to maximize the benefit of objects in a knapsack without exceeding its capacity. It is an NP-complete problem and as such an exact solution for a large input is practically impossible to obtain. The main goal of the studies was to present a comparative study of the brute force, dynamic programming, memory functions, branch and bound, greedy, and genetic algorithms. The study discussed the complexity of each algorithm in terms of time and memory requirements, and in terms of required programming efforts. The author's experimental results showed that the most promising approaches are dynamic programming and genetic algorithms. The study examines in more details the specifics and the limitations of these two paradigms.

Yunhong and Victor (2008) modeled a budget constrained keyword bidding in sponsored search auctions as a stochastic multiple-choice knapsack problem (S-MCKP) and proposed a

new algorithm to solve SMCKP and the corresponding bidding optimization problem. the authors algorithm selects items online based on a threshold function which can be built/updated using historical data. Their algorithm achieved about 99% performance compared to the offline optimum when applied to a real bidding dataset. With synthetic dataset, its performance ratio against the offline optimum converges to one empirically with increasing number of periods.

Rajeev and Ramesh (1992) presented a new greedy heuristic for the integer knapsack problem. The proposed heuristic selects items in non-increasing order of their maximum possible contribution to the solution value given the available knapsack capacity at each step. The lower bound on the performance ratio for this “total-value” greedy heuristic is shown to dominate the corresponding lower bound for the density-ordered greedy heuristic.

George (1995) proposed the average-case behavior of the Zero–One Knapsack problem, as well as an on-line version. The authors allowed the capacity of the knapsack to grow proportionally to the number of items, so that the optimum solution tends to be $\Theta(n)$. Under fairly general conditions on the distribution, they obtained a description of the expected value of the optimum offline solution which is accurate up to terms which are $o(1)$. The authors then considered a simple greedy method for the on-line problem, which is called Online Greedy and is allowed to use knowledge of the distribution, and shown that the solution obtained by this algorithm differs from the true optimum by an average of $\Theta(\log n)$; in fact, and can determine the multiplicative constant hidden by the Θ -notation. Thus on average the cost of being forced to give answers on-line is quite small compared to the optimum solution.

The constrained compartmentalized knapsack problem is an extension of the classical integer

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

The Multiple Knapsack Problem (MKP) is a NP-hard combinatorial optimization problem in many real-world applications. An algorithm with the behaviors of preying, following and swarming of artificial fish for searching optimal solution was proposed by Ma Xuan (2009). With regard to the problem that infeasible solutions are largely produced in the process of initializing individuals and implementing the behaviors of artificial fish due to the multiple constraints, which undermines the algorithm performance, an adjusting operator based on

heuristic rule was designed to ensure all the individuals in the feasible solution areas. Computational results show that the algorithm can quickly find optimal solution. The proposed algorithm can also be applied to other constrained combinatorial optimization problems. The above literature shows that knapsack is a very important tool which has helped in many field.

KNUST



CHAPTER 3

METHODOLOGY

INTRODUCTION

This chapter provides an explanation of the branch-and-bound algorithm for solving our

problem. Before our introduction of the branch-and-bound for knapsack problems, we first give a general overview of the branch-and-bound for general integer programming.

Formally, an integer programming problem is formulated as

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

$$\text{Subject to: } Ax \leq b$$

$$x \geq 0 \text{ and integer.}$$

Where; A

KNUST



either branch and bound or cutting planes can be used to find the solution to the integer programming problem.

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

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

An alternative to the branch and bound method is to use cutting planes to reduce the linear relaxation space. The basic idea of the cutting plane method is to cut off parts of the feasible

region of the corresponding linear program, so that the optimal integer solution becomes an extreme point and can be found by the simplex algorithm. This method attempts to find a hyper plane that intersects the solution space below the current linear relaxation point without eliminating any integer solutions. Once such a hyper plane has been put in place, a new linear relaxation point is found, and branch and bound can be implemented or additional cutting planes can be added until an integer solution is returned as the solution to the linear programming problem.

3.1 Branch-and-Bound Algorithms for Knapsacks

The first branch-and-bound approach to the exact solution of KP was presented by Kolesar (1967). The algorithm consists of a highest-first binary branching scheme with:

- (a) at each node, selects the not-yet-fixed item j having the maximum profit per unit weight and generates two descendent nodes by fixing x_j , respectively, to 1 and 0;
- (b) continues the search from the feasible node for which the value of upper bound U_1 is a maximum

The large computer memory and time requirements of the Kolesar algorithm were greatly reduced by the Greenberg and Hegerich (1970) approach, differing in two main respects:

- (a) at each node, the continuous relaxation of the induced sub problem is solved and the corresponding critical items \hat{s} is selected to generate the two descendent nodes (by imposing X

last node generated by imposing $X_j = 1$, i.e. the algorithm is of depth – first type.

Horowitz and Sahni (1997) (and independently, Ahrens and Finke (1975)) derived from the previous scheme on depth-first algorithm in which;

- (a) selection of the branching variable X_j is the same as in Kolesar;
- (b) the search continues from the node associated with the insertion of item j (condition $X_j = 1$), i.e. following a greedy strategy

Other algorithms have been derived from the Greenberg – Hegerich approach (Barr and Ross (1975), Lauriere (1978)] and from different techniques (Lageireg and Lenstra (1972), Guignard and Spielberg (1972), Fayard and Plateau (1975), Veliev and Mamedov (1981). The Horowitz – Sahni one is, however, the most effective, structured and easy to implement and has constituted the basis for several improvements, including that of Martello – Toth algorithm (Martello and Toth, 1977), which is generally considered highly effective. Hence, we will also restrict our research work to that of the Horowitz and Sahni algorithm and Martello and Toth algorithm.

3.2 The Horowitz – Sanni Algorithm

Assume that the items are sorted. A forward move consists of inserting the largest possible set of new consecutive items into the current solution. A backtracking move consists of removing the last inserted item from the current solution. Whenever a forward move is exhausted, the upper bound U_1 corresponding to the current solution is computed and compared with the best solution so far, in order to check whether further forward moves could lead to a better one; if so, a new forward move is performed, otherwise a backtracking follows. When the last item has been considered, the current solution is complete and

possible updating of the best solution so far occurs. The algorithm stops when no further backtracking can be performed. In the following description of the algorithm we use the notations.

KNUST



begin

KNUST



KNUST



insertion into the current solution of any item following the i th. The procedure is based on the following consideration;

The current solution could be improved only if the i^{th} item is replaced by an item having greater profit and a weight small enough to allow its insertion, or by at least two items having global weight not greater than $W_i +$

KNUST



KNUST



KNUST



KNUST



KNUST



CHAPTER 4

DATA COLLECTION AND ANALYSIS

INTRODUCTION

In this chapter, we shall consider a computational study of branch-and-bound algorithm applied to knapsack instance. Consideration is given to the 0-1 knapsack problem where n

KNUST



Table 4.1: List of the capacity (tons/week) and the cost (1000/unit) for each site

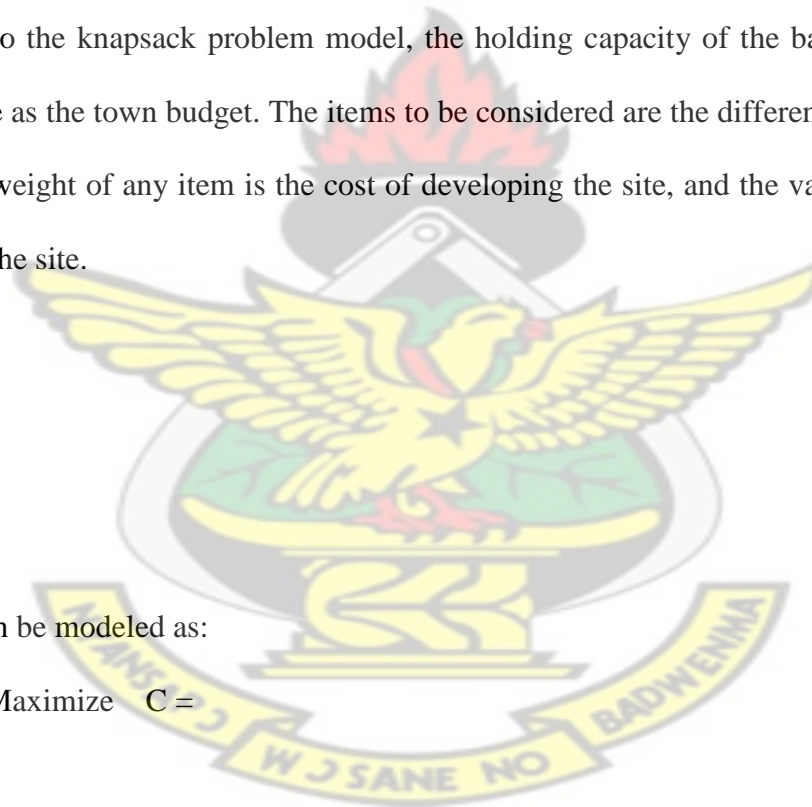
SITE	A	B	C	D	E	F	G
CAPACITY	70	20	39	37	7	5	10
COST	31	10	20	19	4	3	6

The problem here is to select land in such a way that the optimal capacity would be achieved without over shooting the amount allocated for the land development.

In comparison to the knapsack problem model, the holding capacity of the bag is the resource limit, given here as the town budget. The items to be considered are the different sites that can be developed, the weight of any item is the cost of developing the site, and the value of the item is the capacity of the site.

The problem can be modeled as:

Maximize $C =$



w_i = Cost of developing a site

W = Total amount available for development (resource limit)

Thus,

$$\text{Maximize } C = 70s_1 + 20s_2 + 39s_3 + 37s_4 + 7s_5 + 5s_6 + 10s_7$$

$$\text{Subject to: } 31s_1 + 10s_2 + 20s_3 + 19s_4 + 4s_5 + 3s_6 + 6s_7$$

KNUST



6	(1,1,0,0,1,1,0)	102	48
7	(1,1,0,0,1,1,0)	102	48
8	(1,1,0,0,1,0,0)	97	45
9	(1,1,0,0,1,0,0)	97	45
10	(1,1,0,0,0,0,0)	90	41
11	(1,1,0,0,0,1,0)	95	44
12	(1,1,0,0,0,1,1)	105	50
13	(1,1,0,0,0,0,0)	90	41
14	(1,0,0,0,0,0,0)	70	31
15	(1,0,0,0,0,0,0)	70	31
16	(1,0,0,1,0,0,0)	Infeasible	51
17	(1,0,0,1,0,0,0)	Infeasible	51
18	(1,0,0,1,0,0,0)	Infeasible	51
19	(1,0,0,1,0,0,0)	Infeasible	51
20	(1,0,0,0,0,0,0)	70	31

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

INTRODUCTION

We have described the site development problem of a town as a 0-1 knapsack programming problem. We applied the branch-and-bound algorithm of The Horowitz – Sahni to solve the town’s site development problem. Our research focused on the use of the Knapsack problem for site development given a limited available fund for a town in Ghana, but can however be applied to any situation that can be modeled as a 0-1 knapsack problem.

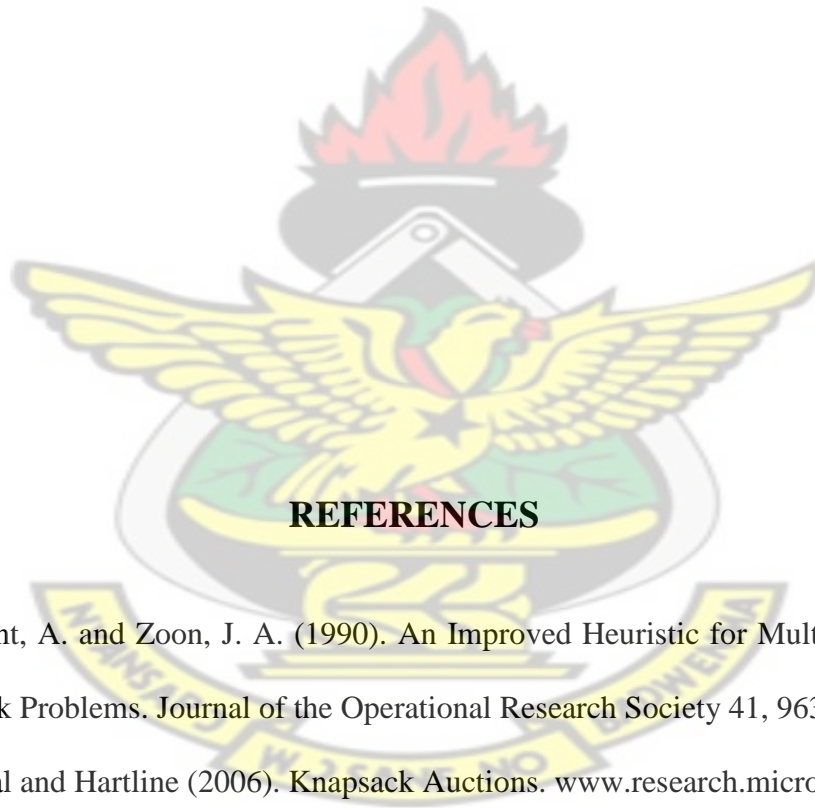
5.1 CONCLUSIONS

This thesis seeks to solve a real-life problem of site development for refuse disposal of Sekondi-Takoradi Metropolis as a knapsack problem using the branch-and-bound algorithm of The Horowitz–Sahni. It was observed that the solution that gave maximum achievable value was (1, 1, 0, 0, 0, 1, and 1). This means that the company should spend a total cost of fifty thousand Ghana cedis (GH¢50,000) to obtain an optimal site development of one hundred and five thousand (105,000) tones per week, consisting of selecting Site A, Site B, Site F, and Site G.

5.2 RECOMMENDATIONS

The use of computer application in computation gives a systematic and transparent solution as compared with an arbitrary method. Using the more scientific Knapsack problem model for the site development of the town's refuse disposal management gives a better result. Management may benefit from the proposed approach for site development for refuse disposal to guarantee optimal refuse disposal capacity in tones per week. We therefore recommend that the Knapsack problem model should be adopted by the metropolitan and district assemblies for refuse disposal management. We also recommend that future study may be conducted to cover the entire country.

KNUST



REFERENCES

1. Volgenant, A. and Zoon, J. A. (1990). An Improved Heuristic for Multidimensional 0-1 Knapsack Problems. *Journal of the Operational Research Society* 41, 963–970.
2. Aggarwal and Hartline (2006). Knapsack Auctions. www.research.microsoft.com
3. Caprara, A., Pisinger, D. and Toth, P.(2007). Exact Solution of the Quadratic Knapsack Problem. *Journals of Operations Research* 55:1001-1021
4. Ram, B. and Sarin, S. (1988). An Algorithm for the 0-1 Equality Knapsack Problem. *Journal of the Operational Research Society* 39, 1045–1049.

5. Bertsimas D., Darnell, C. and Soucy, R. (1999). Portfolio construction through mixed-integer programming at Grantham, Mayo, Van Otterloo and Company. *Interfaces* 29, n1, Jan. – Feb. 1999, 49-66.
6. Sung-Ho, C. (1998). Tactical-Level Resource allocation procedure for the hotel industry. *Journals of Texas A & M Industrial and Systems Engineering*.
7. Christopher, C. A. (1998). Solving Geometric Knapsack Problems using Tabu Search Heuristics.
<http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier>
8. Claessens T., Van Dijk, N. and Zwaneveld P.J. (1998). Coast optimal allocation of rail passenger lines. *European Journal of Operational Research* 110, 474
9. Das, S. and Ghosh, D. (2003). Binary knapsack problems with random budgets . *Journal of the Operational Research Society*
10. Dizdar, D., Gershkov, A. and Moldovanu, B. (2010). Revenue maximization in the dynamic knapsack problem. *Theoretical Economics* 6 (2011), 157–184
11. Der-Chyuan, L. and Chin-Chen, C. (1997). *International Journal of High Speed Computing (IJHSC)*. Change in behaviour of outputs generated on varying the crossover and mutation rates.
12. Doprado M. F. and Marcos, N. A.(2007).The constrained compartmentalised knapsack problem. *Journals of Computers and Operations research*
13. Easton, K., Nemhauser, G., and Trick, M. (2003). Solving the travelling tournament problem, a combined integer programming and constraint programming approach (practice and theory of automated timetabling IV). 4th International conference, PATAT 2002, Lecture notes in computer science vol. 2740, pp. 100-109

- 14 Lawler, E. L. (1977). Fast approximation algorithms for knapsack problems. *Focs*, pp.206-213 18th Annual Symposium on Foundations of Computer Science.
- 15 Hadjiconstantinou, E. and Christofides, N. (2010). An exact algorithm for general, orthogonal, two-dimensional knapsack problems. *European Journal of Operational Research*
- 16 Arkin, E. M., Khuller, S. and Mitchell, J. S. B. (1993). Geometric knapsack problems. <http://www.springerlink.com/content/g007w81p153h3326>
- 17 Geir, D. (1997). An introduction to convexity, polyhedral theory and combinatorial optimization. University of Oslo, Department of Informatics
- 18 Lueker, G. S. (1995). Average-Case Analysis of Off-Line and On-Line Knapsack Problems. *Journal of Algorithms Volume 29, Issue 2, Pages 277-305*
- 19 Ghoseiri K., Szidarowszky, F. and Asgharpour, M. J. (2004). A multi-objective train scheduling model and solution. *Transportation research part B: Methodological* 38, 927.
- 20 Granmo O. C., Oommen, B. J. Myrer, S. A. and Olsen, M. G. (2007). Learning automated-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation. *IEE Transactions on systems, man and cybernetics, part B (cybernetics)*, 37 n1, 166-175.
- 21 Gutierrez and Maria Talia (2007). Lifting general integer programs. Kansas State University Masters thesis
- 22 Haghani, A. and Shafali, Y. (2002). Bus maintenance systems and scheduling: model formulations and solutions. *Transportation research part A: Policy and Practice*, 36, 453.
- 23 Higgins, A., Kozan, E. and Ferreira, L. (1996). Optimal scheduling of train on a single line track. *Transportation research part B: Methodological*, 38, 927.

- 24 Hillier, F. S. and Lieberman, G. J. (2001). Introduction to Operations research. McGraw-Hill, New York 576-581
- 25 Horowi, E. and Sahni (1974). Computing partitions with applications to knapsack problems. Journal of ACM21, 277-292
- 26 Bryce, H. (2007). Finding adjacent facet-defining inequalities. Kansas State University Masters thesis.
- 27 Tauhidul, I. M. (2009). Approximation algorithms for minimum knapsack problem. Master's degree Thesis, UNIVERSITY OF LETHBRIDGE
- 28 Kalai, R. and Vanderpooten, D. (2006). Lexicographic α -Robust Knapsack Problem <http://ieeexplore.ieee.org/xpl/freeabs>
- Michel, S., Perrot, N. and Vanderbeck, F. (2009). Knapsack problems with setups <http://ieeexplore.ieee.org/xpl/freeabs>
- 29 Karp, R. M. (1972). Reducibility among combinatorial problems. Complexity of computer computations; Plenum Press New York 85-103
- 30 Zhanhong, L. S., Wang, Y. Y. and Wei, L. (2010). Streaming Media Caching Model Based on Knapsack Problem. Journal of Networks, Vol 6, No 9 (2011), 1379-1386.
- 31 Xin, L. and Denggu, F. (2004). Quantum algorithm analysis of knapsack problem. JOURNAL OF BEIJING UNIVERSITY OF AERONAUTICS AND A, V 30(11)
- 32 Garg, L. M. and Gupta, S. (2009). An Improved Genetic Algorithm Based on Adaptive Repair Operator for Solving the Knapsack Problem. Journal of Computer Science, volume 5, issue 8, page 544-547
- 33 Mattfeld, D. C. and Kopfer, H. (2003). Terminal operations management in vehicle transshipment. Transportation research part A: Policy and Practice, 37, 435.

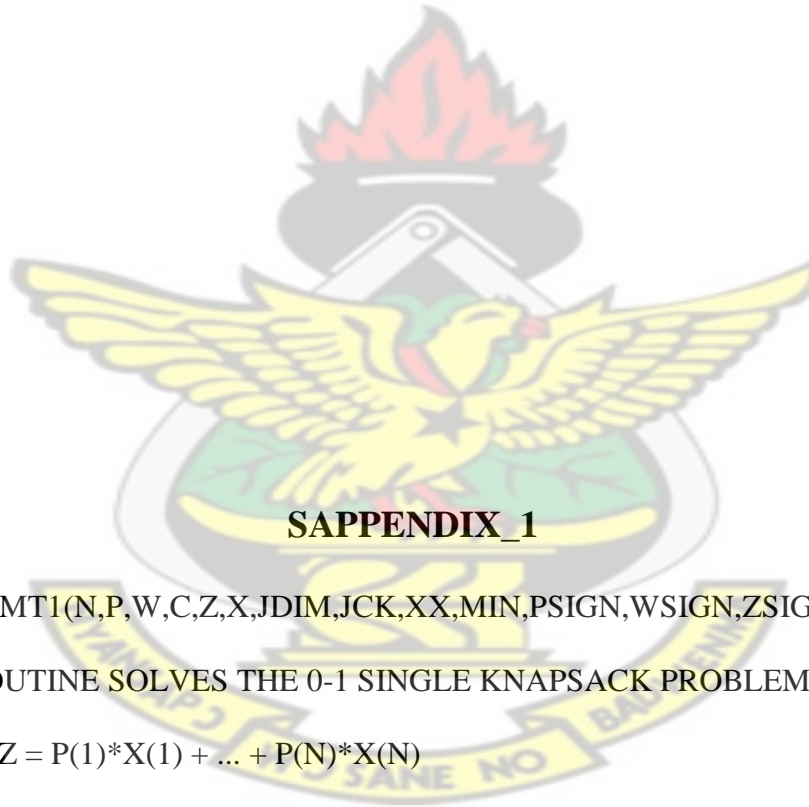
- 34 Xuan, M. A. (2009). Artificial fish swarm algorithm for multiple knapsack problem. *Journal of Computer Applications* 2010, 30(2) 469-471
- 35 Hristakeva, M. and Shrestha, D. (2011). Solving the 0-1 Knapsack Problem with Genetic Algorithms. <http://freetechebooks.com/file-2011/knapsack-problem>
- 36 Hristakeva, M. and Shrestha, D. (2005). Different Approaches to Solve the 0/1 Knapsack Problem. http://micsymposium.org/mics_2005/papers/paper102.
- 37 Michel, S., Perrot, N. and Vanderbeck, F. (2009). Knapsack problems with setups <http://ieeexplore.ieee.org/xpl/freeabs>
- 38 Nemhauser, G. L. and L.Wolsey, L. A. (1998). *Integer and Combinatorial Optimization*. John Wiley and Sons, New York.
- 39 On Stochastic Bilevel Programming Problem with Knapsack Constraints. <http://www.kosuch.eu/stefanie/veroeffentlichungen>
- 40 Ofori, O. E. (2009). Optimal resource Allocation Using Knapsack Problems: A case Study of Television Advertisements at GTV. Master's degree thesis, KNUST
- 41 Kohli, R. and Krishnamurti, R. (1992). A total-value greedy heuristic for the integer knapsack problem. *Operations Research Letters* Volume 12, Issue 2
- 42 Mansini, R. and Speranza, M. G. (2009). An Exact Algorithm for the Multidimensional Knapsack Problem <http://ideas.repec.org/a/eee/ejores/v196y2009i3p909-918>
- 43 Shang, R., Ma, W. and Zhang, W (2006). Immune Clonal MO Algorithm for 0/1 Knapsack Problems. *Lecture Notes in Computer Science*, 2006, Volume 4221/2006, 870-878.

- 44 Kosuch, S. and Lisser, A. (2009). On two-stage stochastic knapsack problems. Discrete Applied Mathematics Volume 159, Issue 16
- 45 Khan, S., LI, K. F., Manning, E. G. and Akbar, M. D. M. (2002). SOLVING THE KNAPSACK PROBLEM FOR ADAPTIVE MULTIMEDIA SYSTEMS. <http://studia.complexica.net/Art/RI020108>
- 46 Srisuwannapa, C. and Charnsethikul, P. (2007). An Exact Algorithm for the Unbounded Knapsack Problem with Minimizing Maximum Processing Time. Journal of Computer Science, 3: 138-143.
- 47 Kosuch, S.(2010). An Ant Colony Optimization Algorithm for the Two-Stage Knapsack Problem. <http://www.kosuch.eu/stefanie>
- 48 Kosuch, S., Letournel, M. and Lisser, A. (2009). On a Stochastic Knapsack Problem. Laboratoire de recherche en Informatique, Universite Paris Sud 91405 Orsay Cedex.
- 49 Stefanie Kosuch, S., Le Bodic, P., Leung, J. and Lisser, A. (2009).
- 50 Tomastik, R. N. (1993). The facet ascending algorithm for integer programming problems. Proceedings on the 32nd IEEE conference on decision and control, 3, 2880-2884.
- 51 Boryczka, U. (2006). The influence of Trial representation in ACO for good results in MKP. From Proceeding (505) Advances in Computer Science and Technology
- 52 Pferschy, U., Pisinger, D. and Woeginger, G. J. (1995). Simple but efficient approaches for the collapsing knapsack problem. Journals of Operations Research.
- 53 Ravi, V. G. R. (2003). Chance Constrained Knapsack Problem with Random Item Sizes. http://www.columbia.edu/~vg2277/stoch_knapsack

54 Yan, S. and Chen, H. L. (2002). A scheduling model and a solution algorithm for inter-city bus carriers. *Transportation research part A: Policy & Practice*, 36, 805.

55 Zhou, Y. and Naroditskiy, V. (2008). Algorithm for Stochastic MultipleChoice Knapsack Problem and Application to Keywords Bidding.
<http://research.yahoo.com/workshops/troa-2008/papers/submission>

KNUST



SAPPENDIX_1

```
SUBROUTINE MT1(N,P,W,C,Z,X,JDIM,JCK,XX,MIN,PSIGN,WSIGN,ZSIGN)
```

```
C THIS SUBROUTINE SOLVES THE 0-1 SINGLE KNAPSACK PROBLEM
```

```
C MAXIMIZE  $Z = P(1)*X(1) + \dots + P(N)*X(N)$ 
```

```
C SUBJECT TO:  $W(1)*X(1) + \dots + W(N)*X(N) \leq C$  ,
```

```
C  $X(J) = 0$  OR  $1$  FOR  $J=1,\dots,N$ .
```

```
C THE INPUT PROBLEM MUST SATISFY THE CONDITIONS
```

```
C 1)  $2 \leq N \leq JDIM - 1$  ;
```

```
C 2)  $P(J), W(J), C$  POSITIVE INTEGERS;
```

C 3) $\text{MAX}(W(J)) \leq C$;

C 4) $W(1) + \dots + W(N) > C$;

C 5) $P(J)/W(J) \geq P(J+1)/W(J+1)$ FOR $J=1, \dots, N-1$.

C MEANING OF THE INPUT PARAMETERS:

C N = NUMBER OF ITEMS;

C P(J) = PROFIT OF ITEM J ($J=1, \dots, N$);

C W(J) = WEIGHT OF ITEM J ($J=1, \dots, N$);

C C = CAPACITY OF THE KNAPSACK;

C JDIM = DIMENSION OF THE 8 ARRAYS;

C JCK = 1 IF CHECK ON THE INPUT DATA IS DESIRED,

C = 0 OTHERWISE.

C MEANING OF THE OUTPUT PARAMETERS:

C Z = VALUE OF THE OPTIMAL SOLUTION IF $Z > 0$,

C = ERROR IN THE INPUT DATA (WHEN JCK=1) IF $Z < 0$: CONDI-

C TION - Z IS VIOLATED;

C X(J) = 1 IF ITEM J IS IN THE OPTIMAL SOLUTION,

C = 0 OR OTHERWISE.

C ARRAYS XX, MIN, PSIGN, WSIGN AND ZSIGN ARE DUMMY.

C ALL THE PARAMETERS ARE INTEGER. ON RETURN OF MT1 ALL THE INPUT

C PARAMETERS ARE UNCHANGED.

INTEGER P(JDIM),W(JDIM),X(JDIM),C,Z

INTEGER XX(JDIM),MIN(JDIM),PSIGN(JDIM),WSIGN(JDIM),ZSIGN(JDIM)

INTEGER CH,CHS,DIFF,PROFIT,R,T

Z = 0

IF (JCK .EQ. 1) CALL CHMT1(N,P,W,C,Z,JDIM)

IF (Z .LT. 0) RETURN

C INITIALIZE.

CH = C

IP = 0

CHS = CH

DO 10 LL=1,N

IF (W(LL) .GT. CHS) GO TO 20

IP = IP + P(LL)

CHS = CHS - W(LL)

10 CONTINUE

20 LL = LL - 1

IF (CHS .EQ. 0) GO TO 50

P(N+1) = 0

W(N+1) = CH + 1

LIM = IP + CHS * P(LL+2) / W(LL+2)

A = W(LL+1) - CHS

B = IP + P(LL+1)

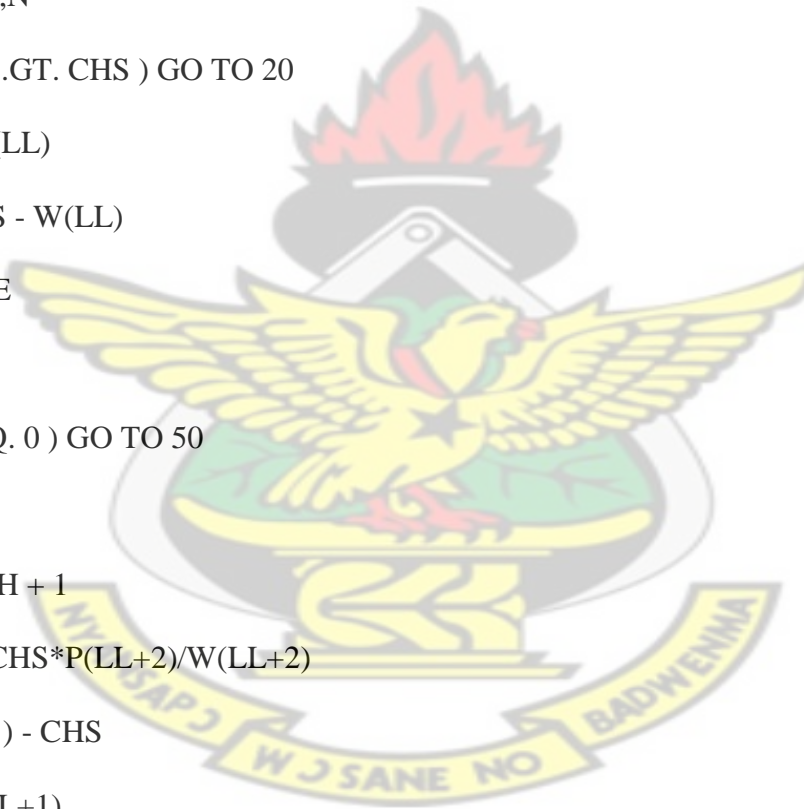
LIM1 = B - A * FLOAT(P(LL)) / FLOAT(W(LL))

IF (LIM1 .GT. LIM) LIM = LIM1

MINK = CH + 1

MIN(N) = MINK

KNUST



DO 30 J=2,N

KK = N + 2 - J

IF (W(KK) .LT. MINK) MINK = W(KK)

MIN(KK-1) = MINK

30 CONTINUE

DO 40 J=1,N

XX(J) = 0

40 CONTINUE

Z = 0

PROFIT = 0

LOLD = N

II = 1

GO TO 170

50 Z = IP

DO 60 J=1,LL

X(J) = 1

60 CONTINUE

NN = LL + 1

DO 70 J=NN,N

X(J) = 0

70 CONTINUE

RETURN

C TRY TO INSERT THE II-TH ITEM INTO THE CURRENT SOLUTION.

KNUST



80 IF (W(II) .LE. CH) GO TO 90

II1 = II + 1

IF (Z .GE. CH*P(II1)/W(II1) + PROFIT) GO TO 280

II = II1

GO TO 80

C BUILD A NEW CURRENT SOLUTION.

90 IP = PSIGN(II)

CHS = CH - WSIGN(II)

IN = ZSIGN(II)

DO 100 LL=IN,N

IF (W(LL) .GT. CHS) GO TO 160

IP = IP + P(LL)

CHS = CHS - W(LL)

100 CONTINUE

LL = N

110 IF (Z .GE. IP + PROFIT) GO TO 280

Z = IP + PROFIT

NN = II - 1

DO 120 J=1,NN

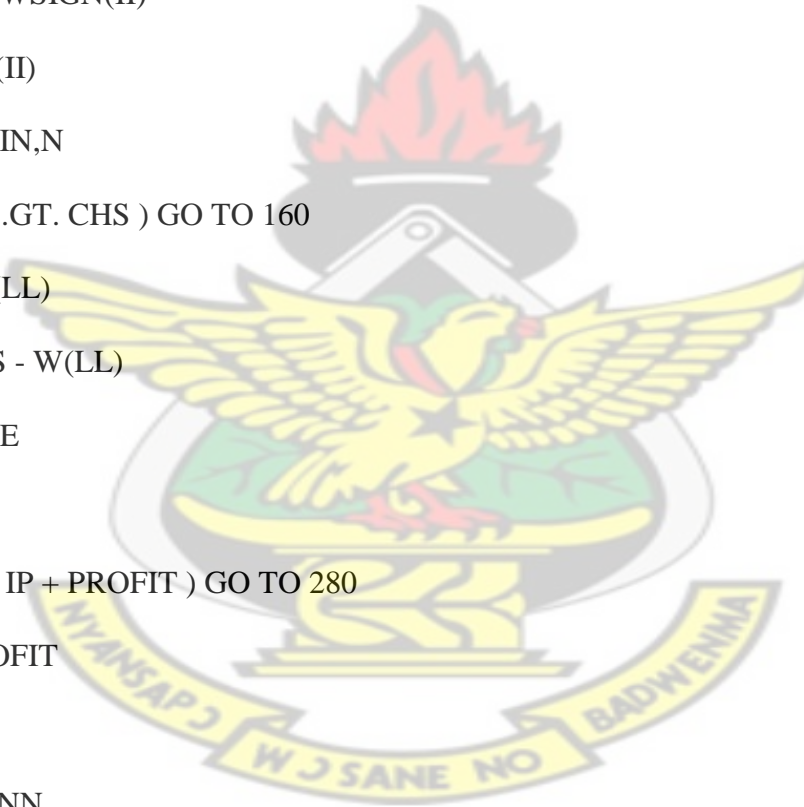
X(J) = XX(J)

120 CONTINUE

DO 130 J=II,LL

X(J) = 1

KNUST



130 CONTINUE

IF (LL .EQ. N) GO TO 150

NN = LL + 1

DO 140 J=NN,N

X(J) = 0

140 CONTINUE

150 IF (Z .NE. LIM) GO TO 280

RETURN

160 IU = CHS*P(LL)/W(LL)

LL = LL - 1

IF (IU .EQ. 0) GO TO 110

IF (Z .GE. PROFIT + IP + IU) GO TO 280

C SAVE THE CURRENT SOLUTION.

170 WSIGN(II) = CH - CHS

PSIGN(II) = IP

ZSIGN(II) = LL + 1

XX(II) = 1

NN = LL - 1

IF (NN .LT. II) GO TO 190

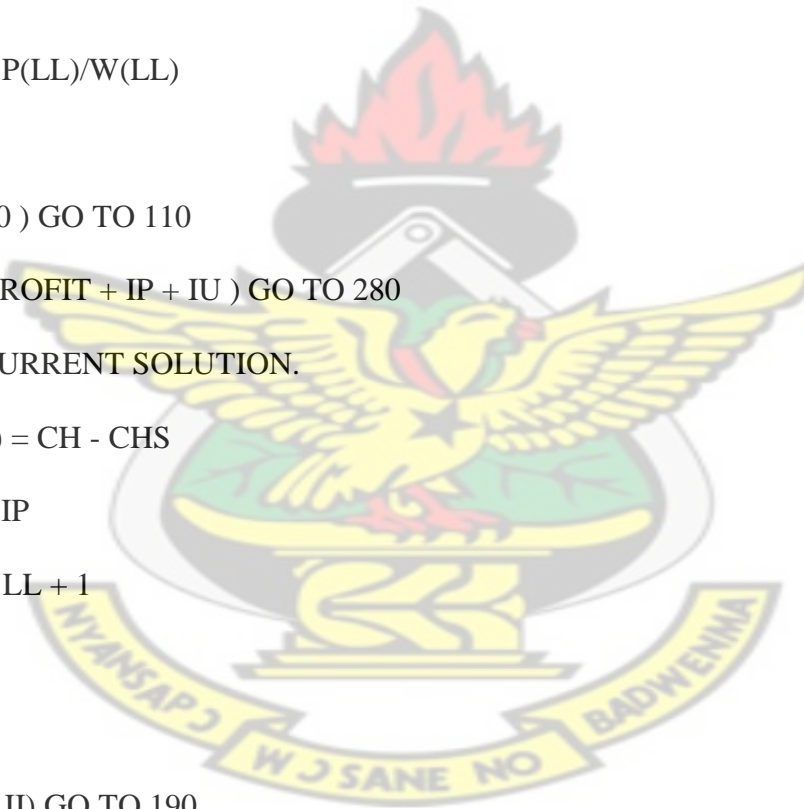
DO 180 J=II,NN

WSIGN(J+1) = WSIGN(J) - W(J)

PSIGN(J+1) = PSIGN(J) - P(J)

ZSIGN(J+1) = LL + 1

KNUST



XX(J+1) = 1

180 CONTINUE

190 J1 = LL + 1

DO 200 J=J1,LOLD

WSIGN(J) = 0

PSIGN(J) = 0

ZSIGN(J) = J

200 CONTINUE

LOLD = LL

CH = CHS

PROFIT = PROFIT + IP

IF (LL - (N - 2)) 240, 220, 210

210 II = N

GO TO 250

220 IF (CH .LT. W(N)) GO TO 230

CH = CH - W(N)

PROFIT = PROFIT + P(N)

XX(N) = 1

230 II = N - 1

GO TO 250

240 II = LL + 2

IF (CH .GE. MIN(II-1)) GO TO 80

C SAVE THE CURRENT OPTIMAL SOLUTION.

KNUST



250 IF (Z .GE. PROFIT) GO TO 270

Z = PROFIT

DO 260 J=1,N

X(J) = XX(J)

260 CONTINUE

IF (Z .EQ. LIM) RETURN

270 IF (XX(N) .EQ. 0) GO TO 280

XX(N) = 0

CH = CH + W(N)

PROFIT = PROFIT - P(N)

C BACKTRACK.

280 NN = II - 1

IF (NN .EQ. 0) RETURN

DO 290 J=1,NN

KK = II - J

IF (XX(KK) .EQ. 1) GO TO 300

290 CONTINUE

RETURN

300 R = CH

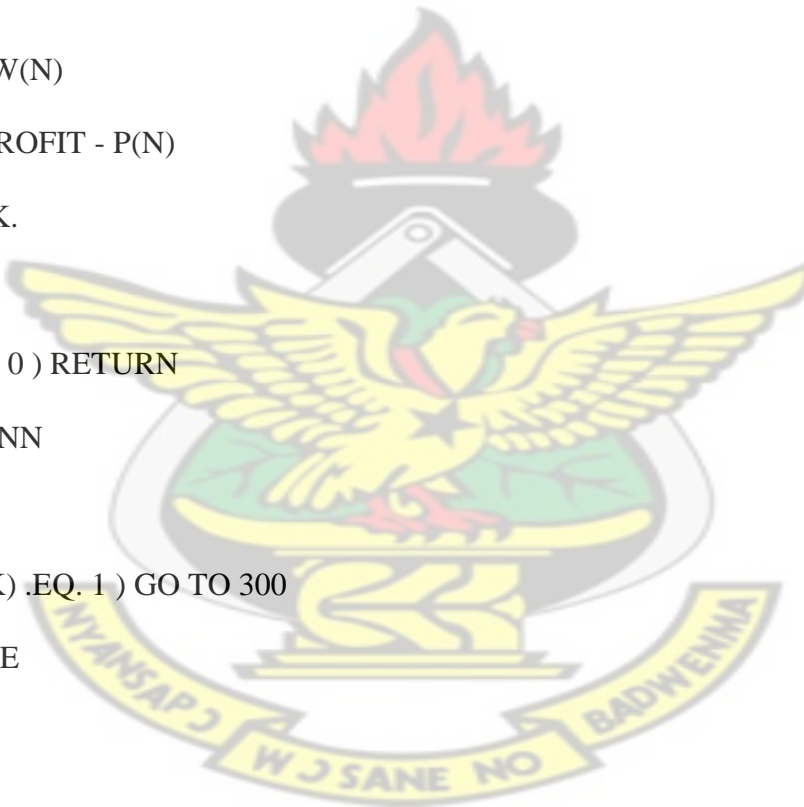
CH = CH + W(KK)

PROFIT = PROFIT - P(KK)

XX(KK) = 0

IF (R .LT. MIN(KK)) GO TO 310

KNUST



II = KK + 1

GO TO 80

310 NN = KK + 1

II = KK

C TRY TO SUBSTITUTE THE NN-TH ITEM FOR THE KK-TH.

320 IF (Z .GE. PROFIT + CH*P(NN)/W(NN)) GO TO 280

DIFF = W(NN) - W(KK)

IF (DIFF) 370, 330, 340

330 NN = NN + 1

GO TO 320

340 IF (DIFF .GT. R) GO TO 330

IF (Z .GE. PROFIT + P(NN)) GO TO 330

Z = PROFIT + P(NN)

DO 350 J=1, KK

X(J) = XX(J)

350 CONTINUE

JJ = KK + 1

DO 360 J=JJ, N

X(J) = 0

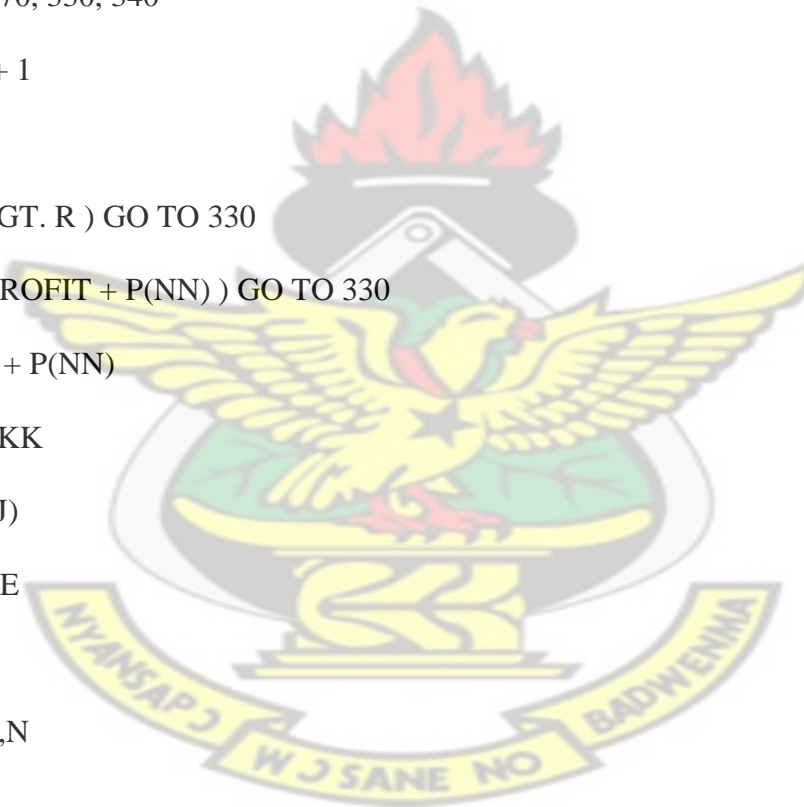
360 CONTINUE

X(NN) = 1

IF (Z .EQ. LIM) RETURN

R = R - DIFF

KNUST



KK = NN

NN = NN + 1

GO TO 320

370 T = R - DIFF

IF (T .LT. MIN(NN)) GO TO 330

IF (Z .GE. PROFIT + P(NN) + T*P(NN+1)/W(NN+1)) GO TO 280

CH = CH - W(NN)

PROFIT = PROFIT + P(NN)

XX(NN) = 1

II = NN + 1

WSIGN(NN) = W(NN)

PSIGN(NN) = P(NN)

ZSIGN(NN) = II

N1 = NN + 1

DO 380 J=N1,LOLD

WSIGN(J) = 0

PSIGN(J) = 0

ZSIGN(J) = J

380 CONTINUE

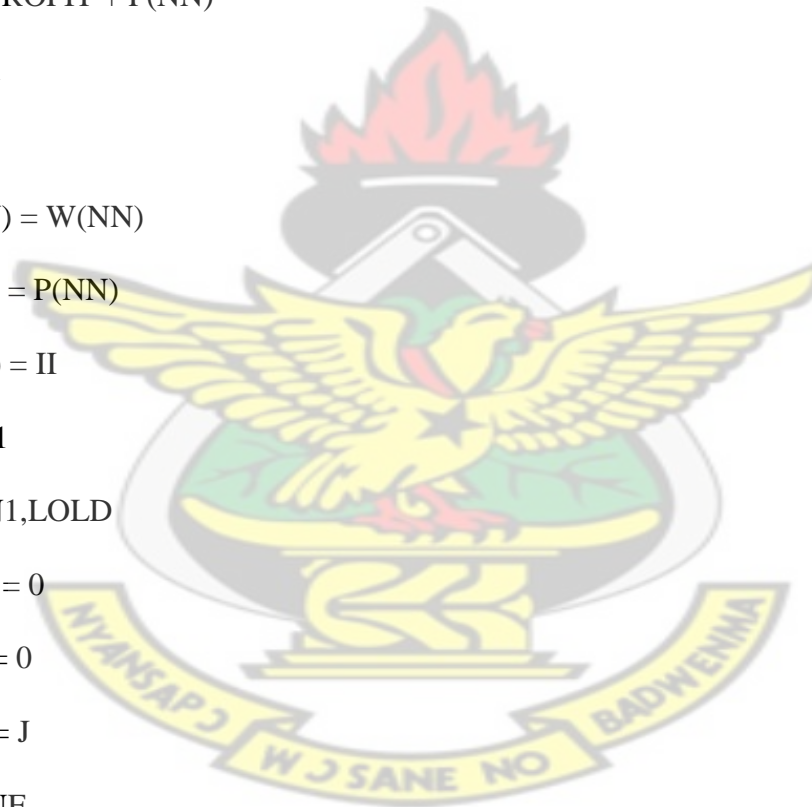
LOLD = NN

GO TO 80

END

SUBROUTINE CHMT1(N,P,W,C,Z,JDIM)

KNUST



C

C CHECK THE INPUT DATA.

C

INTEGER P(JDIM),W(JDIM),C,Z

IF (N .GE. 2 .AND. N .LE. JDIM - 1) GO TO 10

Z = - 1

RETURN

10 IF (C .GT. 0) GO TO 30

20 Z = - 2

RETURN

30 JSW = 0

RR = FLOAT(P(1))/FLOAT(W(1))

DO 50 J=1,N

R = RR

IF (P(J) .LE. 0) GO TO 20

IF (W(J) .LE. 0) GO TO 20

JSW = JSW + W(J)

IF (W(J) .LE. C) GO TO 40

Z = - 3

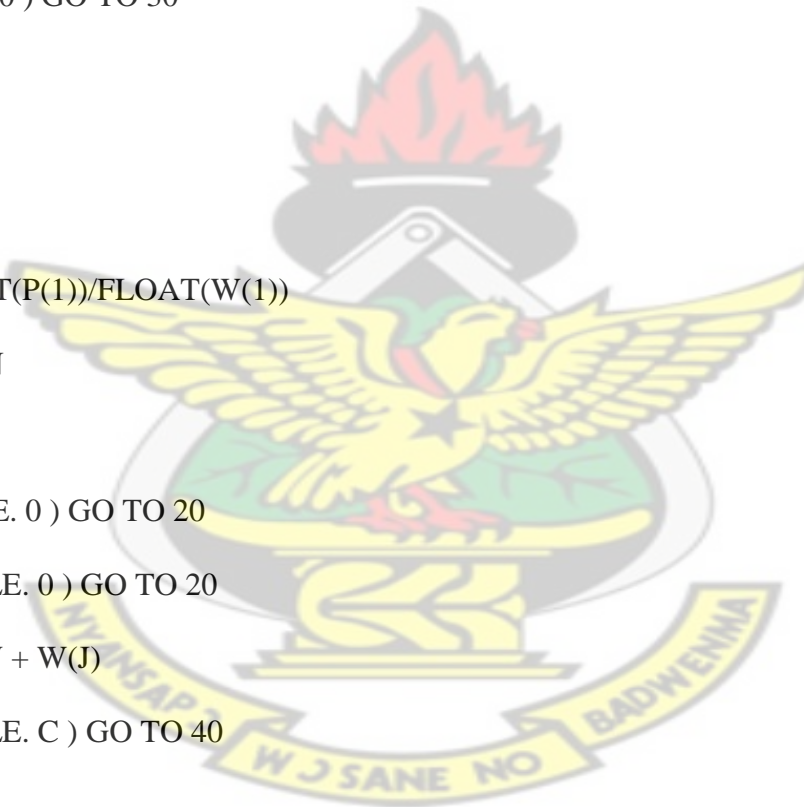
RETURN

40 RR = FLOAT(P(J))/FLOAT(W(J))

IF (RR .LE. R) GO TO 50

Z = - 5

KNUST



RETURN

50 CONTINUE

IF (JSW .GT. C) RETURN

Z = - 4

RETURN

END

KNUST

