# OPTIMAL RESOURCES ALLOCATION; A CASE STUDY OF SEKYERE CENTRAL DISTRICT ASSEMBLY, NSUTA-ASHANTI

By

OWUSU-BEMPAH EMMANUEL (B.Ed Mathematics)



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

MASTER OF PHILOSOPHY (APPLIED MATHEMATICS)

June, 2013

## DECLARATION

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



OWUSU-BEMPAH EMMANUEL . . . . . .

Student (PG6202411) Signature Date

Certified by:

PROF S. K. AMPONSAH . . . . . .

Supervisor Signature Date

Certified by:

PROF. S. K AMPONSAH . . . . . .

Head of Department Signature Date

## ACKNOWLEDGEMENTS

At a distance from the efforts of me, the success of any project depends largely on the support and guiding principle of many others. I seize this opportunity to express my gratitude to the people who have been influential in the triumphant completion of this project. I would like to show my greatest appreciation to my supervisor, Prof. S. K Amponsah. I cannot say thank you enough for his remarkable support and help. I feel motivated and encouraged every time I meet him. Devoid of his back-up and direction this project would not have materialized. The guidance and support received from all loved ones, especially my wife Christiana Owusu-Bempah and my family who contributed to this project, was vital for the success of the project. I am grateful for their constant support and help.

## DEDICATION

This thesis is dedicated to my parents, Mr. and Mrs. Owusu-Bempah who have supported me all the way since the beginning of my studies. Also, this thesis is dedicated to my family who has been a great source of motivation and inspiration. Finally, this thesis is dedicated to Katakyie Kwame Dwumah of the Ghana Revenue Authority, Accra and all those who believe in the richness of learning.



## ABSTRACT

Education is input to eradicate poverty in the recent civilization and this cannot be overemphasized and however, the financial outlay required to provide this very important service is far from being enough as a lot of schools are under trees. In this thesis, we consider the problem of allocating resource at the Sekyere central District Assembly, Nsuta-Ashanti with the aim of minimizing unnecessary lapses during budget allocation for resources by the assembly. The problem was formulated as an Integer Linear Programming (ILP) problem using the available data from the District Assembly. This problem was solved with the Branch and- Bound of Method of solving Integer Linear Programming (ILP). It was found that the out of the ten different locations considered and budget of Three Hundred and Sixty Thousand Ghana Cedis, the optimal number of classroom to be built was fifteen (15) representing a 3-unit classroom and two 6-unit classroom buildings at three different locations within the District at a minimum budget of Three Hundred and Seventeen Thousand Ghana Cedis (GH¢ 317,000) respectively. We concluded that the Knapsack problem for selecting required sites in critical situations such as construction of school buildings was useful and it can be applied to any situation where allocation of funds in the sector of educational development becomes a serious setback. Scientific modeling of allocating resources was recommended as it can be used to reduce financial loss in budget allocations.

## TABLE OF CONTENTS

## LIST OF TABLES

# CHAPTER ONE

## 1.0 INTRODUCTION

It is a universal truth that education is key to eradicating poverty in the modern society and this cannot be overemphasized. In a developing country like Ghana education will help citizens to acquire the needed skills and knowledge. The skill and knowledge acquired will make the citizens functionally literate and productive to facilitate poverty alleviation and promote the rapid socio-economic growth. The government of Ghana embarked on the Basic Education Sector Improvement Program and more popular the Free Compulsory and Universal Basic Education Program (BESIP/FCUBE), which was aimed at providing every child of school-going age with good basic education. However, the financial outlay required to provide this very important service is far from being enough as a lot of schools are under trees. Ghana Education Trust fund was established in September 2000 by the government in providing educational finance supplementation particularly at the basic and secondary level where there exists an urgent need to provide adequate classroom infrastructure for our rapidly growing in-school population. (http://www.getfund.gov.gh). The limited funds allocated to the various Metropolitan, Municipal and District Assemblies should be efficiently used.

The problem that frequently arises in resource allocation where there are financial constraints can be considered as a knapsack problem. This study seeks to provide a scientific way of allocating funds in the building of unit classroom that will ensure it optimality using linear integer programming with Sekyere Central District as a case study.

## 1.1 BACKGROUND OF STUDY

Education is a fundamental human right for all children and this right may not be realized in Ghana if strategic measures are not put in place to ensure adequate infrastructure provision to schools, especially in rural communities. School infrastructure is everything from electricity, toilets, safe buildings, tables, chairs, libraries, computer rooms, safe classrooms, sports fields, laboratories for science experiments, running water and fencing. It is vital when we consider the fact that school infrastructure or resources, impact on how well teachers are able to teach and learners are able to learn. Learners attending schools with better infrastructure tend to perform better than learners who come from schools under trees. Meanwhile, the poor state of school infrastructure was evident in the number of public schools under trees. Lack of appropriate infrastructure for KinderGartens (KGs), poorly ventilated classrooms, poor lighting in classrooms and lack of sanitary facilities for boys and girls among others. It is expected that all stakeholders particularly Civil Society, government, District, municipal and metropolitans assemblies ensure that funds provided are put into proper use. Most of the newly created district lacked school infrastructure that will ensure smooth teaching and learning process.

The government allocated funds for putting up unit classrooms in these newly created districts. This calls for a scientific way or method that will help in the allocation of the provided fund in the Sekyere district in putting up schools in the community. The district has allocated some fund to build unit classroom and must decide on which of these communities to put up the structures. Modern society, with advanced technology usually needs to make best possible decisions, which example involve the best possible

use of resources or funds allocated to the educational sector to minimize production or guarantee full benefit of all.

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).

The knapsack problem has been studied for more than a century, with early works dating as far back as 1897 for the reason that their direct application to problems arises in industries and also for their contribution to the solution methods for integer programming problems. Quite a lot of exact algorithms based on branch and bound,

dynamic programming and heuristics have been proposed to solve the Knapsack Problems.

## 1.2 PROBLEM STATEMENT

The future of nation depends on the type of quality education. Education cannot be reliable when there are limited educational infrastructures. This hinders the progress fighting illiteracy inn such nations. This has been a problem faced by the Sekyere central district since it inception as a newly created district. However, putting up sufficient school buildings within the district will help fight the high rate of illiteracy. Therefore, there is the need to put up maximum number of unit classrooms at affected towns at minimum cost within the district.

The thesis seeks to use Knapsack Problem (KP) to help solve this problem. The knapsack problem is an all-purpose resource allocation problem in which a single resource is assigned to a number of alternatives with the objective of maximizing the total return. The problem is a distribution of effort problem that has a linear objective function and a single constraint.

## 1.3 OBJECTIVES

The objectives of the study are:

(i) to model a real-life problem in developing sites for unit classroom as a 0-1 knapsack problem, and propose branch-and-bound algorithm.

(ii) to determine the maximum number of unit classrooms required to be built on selected sites.

## 1.4 JUSTIFICATION OF THE STUDY

Funds are provided to the various Metropolitan, Municipal and District Assemblies within the country to carry out needed developmental projects. Most times, the Assemblies have to generate funds to initiate project from available revenues. Since the funds made available to undertake project are in limited supply there is the need to put it into judiciously to realize it maximum benefit. Needed attention should therefore be given to areas like educational, health and other social projects. Without any adequate scientific method of selecting from numerous to undertake, the maximum returns from these may not be achieved. Several factors about projects to be undertaken are considered especially location and cost. A number of practical problems can be formulated for example the simple capital budgeting problem of choosing which project constraint on total cost. The Branch and Bound method can be applied to model such managerial and industrial situations.



## 1.5 METHODOLOGY

The study seeks to use the branch and bound method in finding an appropriate solution to the formulated knapsack problem of the number of sites that will give a maximum unit classroom at a minimum cost. Knapsack Problems are among the simplest integer programming problems which are NP-hard. The classical 0-1 Knapsack Problem arises when there is one knapsack and one item of each type. The this study the knapsack Problem was formulated as selecting from a ten site for the construction of unit classrooms given a specified amount of budgeted money. The set of unit classroom selected with maximum room should have a minimum cost out of the lot and not

exciding the budget. All of this will be achieved by using software called quantitative management which helps in solving and analyzing.

## 1.6 SCOPE OF THE STUDY

This study is within the confines of the Sekyere Central District with Nsuta as its capital in the Ashanti Region. We believed that since its newly created district there is the need to improve the educational infrastructure to meet the increasing population. The study is however limited to the data on sites accessed from the budget office of the district assembly.



## 1.7 LIMITATIONS OF THE STUDY

There are a several factors, which limit this study to some level of accuracy. One factor which limits this study is the access to required information from the office of the Sekyere Central District Assembly. Another factor to be considered is the time span which this research is to be carried. A continuous study has to be made thoroughly before an accurate and perfect could be achieved.

## 1.8 ORGANIZATION OF THE STUDY

The study is organized into five chapters. In chapter one, we presented the background, problem statement, objective, justification, methodology and limitation of the study. In chapter two, relevant literature in the field of Knapsack problems and it variants will be discussed. In chapter three, the branch-and-bound algorithm would be introduced and explained. Chapter four provides a computational study the branch-and-bound algorithm

applied to knapsack instances. Chapter five concludes this thesis with additional comments on branch-an-bound algorithm.

## 1.9 SUMMARY

Integer programming, which is an important class of mathematical programming problems, is a useful tool for modeling 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. The Knapsack Problems are widely used in financial decision making, and very interesting from the perspective of computer science since they are NP-complete. These essentials make the studies of Knapsack Problems and their algorithms an extremely important area of research. This thesis seeks to model a real-life problem in site growth for development projects as a 0-1 knapsack problem, and propose branch-and-bound algorithm for solving the problem.

In the next chapter, we shall put forward adequate and relevant literature on Knapsack Problem and its variants.

# CHAPTER TWO

## LITERATURE REVIEW

### 2.0 INTRODUCTION

This chapter presents a review of relevant literatures on Knapsack Problem and its applications. The Knapsack Problem is a problem in combinatorial optimization. Given the set of items, each with a weight and value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. The problem often arises in resource allocation where there are financial constraints.

### 2.1 RELEVANT LITERATURES

Knapsack problems are 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) considered 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.

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) offered 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.



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 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 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 Quadratic Knapsack Problem (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 four hundred (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. The authors 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.



Several exact algorithms based on branch and bound, dynamic programming and heuristics have been proposed to solve the Knapsack Problems. 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.

Oppong (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 2h. 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 of 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. 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. The authors 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 Langrangean 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 which 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 $0(n + 1/\varepsilon 3)$ space. The time bound is reduced to $0(n + 1/\varepsilon 3)$ for the "unbounded" knapsack problem. For the "subset-sum" problem, $0 (n + 1/\varepsilon 3)$ times and $0 (n + 1/\varepsilon 2)$ spaces, or $0(n + 1/\varepsilon 2 \log (1/\varepsilon))$ time and space, are achieved. The "multiple choice" problem, with m equivalence classes, is solved in $0(nm2/\varepsilon)$ time and space.

Lawler (1997) presented fully polynomial approximation algorithms for which 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 $0(n + 1/\varepsilon 3)$ space. The time bound is reduced to $0(n + 1/\varepsilon 3)$ for the "unbounded" knapsack problem. For the "subset-sum" problem, $0 (n + 1/\varepsilon 3)$ times and $0 (n + 1/\varepsilon 2)$ spaces, or $0(n + 1/\varepsilon 2 \log (1/\varepsilon))$ time and space, are achieved. The "multiple choice" problem, with m equivalence classes, is solved in $0(nm2/\varepsilon)$ 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. The authors 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: (i) in each step, not one, but more multiplier values are computed simultaneously, and (ii) 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 PTAS for the case when each item size is normally distributed and independent of other items. They also presented a parametric LP formulation and show 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 $(1+\epsilon)$-factor of the objective value of the fractional solution for any $\epsilon > 0$.

The dominant traffic on the Internet has changed from text and graphics based Web content to more information-rich streaming media content, such as audio and video. With the dramatic increase of network bandwidth and the advancement of technologies

on media authoring, encoding, and distribution, media traffic on the Internet has increased explosively and now accounts for the majority of traffic volume. Modern Internet streaming services have utilized various techniques to improve the quality of streaming media delivery. Proxy server is one of the main solutions used to improve Internet QoS, especially for the QoS of streaming media.

Replacement algorithm optimization is the core of caching model research. However, existing techniques for caching text and image resources are not appropriate for the rapidly growing number of continuous media streams. Based on the concept of hit ratio, Lei Shi et al., (2010) presented a 0-1 knapsack problem that set up a hit ratio model of proxy cache, by use of which a proxy cache policy is presented. As compared with the classical dynamic streaming scheduling strategies, the proposed algorithm is shown that it can make full use of space of proxy cache, and also get a higher hit ratio.

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. Karnin (1984), 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 $O(2^{2n/3})$.

Lou and 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, the 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.

During last few decades, Knapsack problem has been studied through different approaches, according to the theoretical development of combinatorial optimization. Garg and Sunanda (2009) studied 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 antibodied. Meanwhile, the non-dominated antibodies are allowed to survive and to clone. A metric of Coverage of Two Sets is 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. The authors 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.

Zhang et al., (2004), saw 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.

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 technique 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, bases 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 reduces to the famous secretary problem which was first introduced by Dynkin (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 matriod secretary problem in which the elements of a weighted matriod 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 maxi mized, without exceeding the capacity of each of the knap sacks. 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 are 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 then 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 propose a scaling based fully polynomial time approximation scheme for the minimum knapsack problem. The author compared the performance of

this algorithm with existing algorithms. 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. His experiments show 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.

The Multiple Knapsack Problem (MKP) is a NP-hard combinatorial optimization problem in many real-word 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.

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) studied** 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 Θ (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 Θ(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.

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 ten (10) benchmark problems of sizes up to one hundred and five (105) and for seven classical problems of sizes up to five hundred (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.

Elhedhli (2005) considered a class of nonlinear knapsack problems with applications in service systems design and facility location problems with congestion. They provided two linearization 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.

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).

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.

Witzgall (1975) presented a problem which arises in telecommunications when a number of sites for satellite stations have to be selected, such that the global traffic between these stations is maximized and a budget constraint is respected. This problem appears to be a QKP. Similar models arise when considering the location of airports, railway stations or freight handling terminals.

Johnson et al (1993) considered the graph version of the Quadratic Knapsack Problem. After linearization of the objective function, the model is solved by a branch-and-cut system in which tree inequalities and star inequalities are used to tighten the formulation.

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.

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.

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.



## 2.3 SUMMARY

In this chapter, we reviewed relevant and adequate literature on Knapsack Problems. The next chapter presents the research methodology of the study.

# CHAPTER THREE

## METHODOLOGY

### 3.0 INTRODUCTION

The fundamental theory of Linear Programming with regards to its definition and formulation, component, objectives and the method of analysis of the current data to arrive at the objective will be discussed in this chapter.



### 3.1 PROFILE OF THE SEKYERE CENTRAL DISTRICT

The Sekyere Central District with Nsuta as its capital is geographically located within longitude 0.05$^{o}$ and 1.30$^{o}$ W and latitudes 6.55$^{o}$ and 7.30$^{o}$N in the Ashanti Region of Ghana. It was created by the former President, J.A Kufuor which was inaugurated on the 29th of February 2008. It covers a total land area of about 1,564sq. km. Have about 105 settlements with about 70% being rural.

The district is generally low lying and gradually rising through rolling hills stretching southward towards Nsuta. The highest point is 2400m whilst the lowest is 135m above mean sea level. It is fairly drained by several streams and rivers like Afram, Sene, Sasebonso, and Kyirimfa. The Sekyere Central district appears to be doing well in education, despite the numerous problems of the educational sub-sector. Effort should therefore be made to put up more school buildings to enhance the basic education and to make education the basic right for all children.The Sekyere Central district is having limited number of school buildings since it is among the newly created district in the

Ashanti Region of Ghana. This has really made it difficult for populace within its rural location to have access to basic education. The lack of school buildings within the district could lead to severe dropouts in the school attendance in the district. Even though the District Assembly has been able to construct a number of schools infrastructures in the district a lot needs to be done in that area of infrastructure and Classroom. More schools infrastructures have to be built in the district especially in the rural areas because most rural areas in the district do not have good road access to the towns to have a taste of education in proper school buildings. Moreover, few of the schools need to be given additional facilities to meet the increase in enrollment. The most problematic area is the basic education level.



## 3.2 LINEAR PROGRAMMING

Linear programming is a mathematical method for determining a way to achieve the best outcome (such as maximum profit or lowest cost) in a given mathematical model for some list of requirements represented as linear relationships. Linear programming is a specific case of mathematical programming (mathematical optimization).

More formally, linear programming is a technique for the optimization of a linear objective function, subject to linear equality and linear inequality constraints. Its feasible region is a convex polyhedron, which is a set defined as the intersection of finitely many half spaces, each of which is defined by a linear inequality. Its objective function is a real-valued affine function defined on this polyhedron. A linear

programming algorithm finds a point in the polyhedron where this function has the smallest (or largest) value if such a point exists.

Linear programs are problems that can be expressed in canonical form:

maximize $c^T x$

subject to $Ax \leq b$

and $x > 0$, where x represents the vector of variables (to be determined), c and b are vectors of (known) coefficients, *A* is a (known) matrix of coefficients, and $(.)^T$ is the matrix transpose. The expression to be maximized or minimized is called the objective function ($c^T x$ in this case). The inequalities $Ax \leq b$ is the constraints which specify a convex polytope over which the objective function is to be optimized. In this context, two vectors are comparable when they have the same dimensions. If every entry in the first is less-than or equal-to the corresponding entry in the second then we can say the first vector is less-than or equal-to the second vector.



## 3.3 TERMINOLOGIES IN LINEAR PROGRAMMING

Every linear model consists of a set of decision variable which represents the decisions to be made. This is in contrast to a problem data, which are values that are either given or can be simply calculated from what is given.

### 3.3.1 DECISION VARIABLES

Decision variables describe the quantities that the decision makers would like to determine. They are the unknowns of a mathematical programming model. Typically, it optimum values with an optimization method can be determined. In a general model, decision variables are given algebraic designations such as, $x_1, x_2, x_3, \ldots\ldots x_n$. The number of decision variables is *n*, and $x_j$ is the name of the *j*th variable. In a specific situation, it is often convenient to use other names such as $x_{ij}$ or $y_k$ or $z(i,j)$. An assignment of values to all variables in a problem is called a solution.



### 3.3.2 OBJECTIVE FUNCTION

The objective function evaluates some quantitative criterion of immediate importance such as cost, profit, utility, or yield. The general linear objective function can be written as

$$z = c_1x_1 + c_2x_2 + \cdots + c_nx_n = \sum_{j=1}^{n} c_j x_j$$

Here $c_j$ is the coefficient of the *j*th decision variable. The criterion selected can be either maximized or minimized.

### 3.3.3 CONSTRAINTS

A constraint is an inequality or equality defining limitations on decisions. Constraints arise from a variety of sources such as limited resources, contractual obligations, or physical laws. In general, an LP is said to have *m* linear constraints that can be stated as

$$\sum_{j=1}^{n} a_{ij} x_j \begin{Bmatrix} \leq \\ = \\ \geq \end{Bmatrix} b_i, for\ 1 = \cdots m$$

One of the three relations shown in the large brackets must be chosen for each constraint. The number $a_{ij}$ is called a "technological coefficient," and the number $b_i$ is called the "right-side" value of the *i*th constraint. Strict inequalities (<, >, and ) are not permitted. When formulating a model, it is good practice to give a name to each constraint that reflects its purpose.



### 3.3.4 SIMPLE UPPER BOUND

Associated with each variable, $x_j$ may be a specified quantity, $u_j$, that limits its value from above;

$$x_j \leq u_j, for\ j = 1 \ldots . n$$

When a simple upper is not specified for a variable, the variable is said to be unbounded from above.

### 3.3.5 NON-NEGATIVITY RESTRICTIONS

In most practical problems the variables are required to be nonnegative;

$$x_j \geq 0, for\ j = 1 \ldots n$$

This special kind of constraint is called a non-negativity restriction. Sometimes variables are required to be non-positive or, in fact, may be unrestricted (allowing any real value).

### 3.3.6 COMPLETE LINEAR PROGRAMMING MODEL



Combining the aforementioned components into a single statement gives:

$$maximize\ or\ minimize\ z = \sum_{j=1}^{n} c_j x_j$$

$$subject\ to \sum_{j=1}^{n} a_{ij} x_j \begin{Bmatrix} \leq \\ = \\ \geq \end{Bmatrix} b_i, for\ i = 1 \dots m$$

$$0 \leq x_j \leq u_j\ for\ j = 1 \dots n$$

The constraints, including non-negativity and simple upper bounds, define the feasible region of a problem.

### 3.3.7 PARAMETERS

The collection of coefficients $(c_j, a_{ij}, b_i, u_j)$ for all values of the indices *i* and *j* are called the parameters of the model. For the model to be completely determined all parameter values must be known.

## 3.4 BASIC ASSUMPTIONS OF LINEAR PROGRAMMING

For a problem to be realistically represented as a linear program, the following assumptions should hold:

(i) The constraints and objective function are linear.

(a) This requires that the value of the objective function and the response of each resource expressed by the constraints are proportional to the level of each activity expressed in the variables.

(b) Linearity also requires that the effects of the value of each variable on the values of the objective function and the constraints are additive. In other words, there can be no interactions between the effects of different activities; i.e., the level of activity $X_1$ should not affect the costs or benefits associated with the level of activity $X_2$.

(ii) Divisibility: the values of decision variables can be fractions. Sometimes these values only make sense if they are integers; then we need an extension of linear programming called integer programming.

(iii) Certainty: the model assumes that the responses to the values of the variables are exactly equal to the responses represented by the coefficients.

(iv) Data: formulating a linear program to solve a problem assumes that data are available to specify the problem.

## 3.5 FUNDAMENTAL THEOREM OF LINEAR PROGRAMMING

The fundamental theorem of linear programming, in a weak formulation, states that the maxima and minima of a linear function over a convex polygonal region occur at the region's corners. Further, if an extreme value occurs at two corners, then it must also occur everywhere on the line segment between them.



## 3.6 IMPORTANCE OF LINEAR PROGRAMMING

(i) **Solve the business problems:** with linear programming, business problem can be solved easily. It is much benefited for increase the profit or decreases the cost of business.

(ii) **Select best advertising media:** with linear programming, best advertising media among a numbers of media can be selected.

(iii) **Solve the diet problems:** with linear programming, diet problems can be solved with minimum cost. It is very useful for hospitals .There are different elements like vitamins, proteins, carbohydrates etc. You can select best quantity of them with minimum cost.

(iv) **Use in solving staffing problems:** with linear programming, the number of staff needed in hospitals, mines, hotels and other type of business can be solved.

## 3.7 SIMPLEX METHOD

The simplex method is a method for solving problems in linear programming invented by George Dantzig in 1947. The method tests adjacent vertices of the feasible set in sequence so that at each new vertex the objective function improves or is unchanged.

The simplex method is very efficient in practice, generally taking 2m to 3m iterations at most (where $m$ is the number of equality constraints), and converging in expected polynomial for certain distributions of random inputs. However, its worst-case complexity is exponential, as can be demonstrated with carefully constructed examples. Different types of methods for solving linear programming problems are interior point methods, whose complexity is polynomial for both average and worst case.

These methods construct a sequence of strictly feasible points that converges to the solution.

The latter method solves an unconstrained minimization problem in n dimensions by maintaining each iteration n+ 1 point that defines a simplex. At each iteration, this simplex is updated by applying certain transformations to it so that "rolls downhill" until it finds a minimum. In brief, the simplex method passes from vertex to vertex on the boundary of the feasible polyhedron, repeatedly increasing the objective function until either an optimal solution is found, or it is established that no solution exists. In principle, the time required might be an exponential function of the number of variables, and this can happen in some contrived cases. In practice, however, the method is highly efficient, typically requiring a number of steps which is just a small multiple of the number of variables. Linear programs in thousands or even millions of variables are routinely solved using the simplex method on modern computers. Efficient, highly sophisticated implementations are available in the form of computer software packages.

## 3.8 SUMMARY OF THE SIMPLEX METHOD

A. Add slack variables to change the constraints into equations and write all variables to the left of the equal sign and constants to the right.

B. Write the objective function with all nonzero terms to the left of the equal sign and zero to the right. The variable to be maximized must be positive.

C. Set up the initial simplex tableau by creating an augmented matrix from the equations, placing the equation for the objective function last.

D. Determine a pivot element and use matrix row operations to convert the column containing the pivot element into a unit column.

E. If negative elements still exist in the bottom row, repeat Step 4. If all elements in the bottom row are positive, the process has been completed.

F. When the final matrix has been obtained, determine the final basic solution. This will give the maximum value for the objective function and the values of the variables where this maximum occurs.



### 3.9 SIMPLEX ALGORITHM

The simplex algorithm is a method of solving linear programming problems. It's used to reach a goal while also having constraints. Since this is mathematics, it deals with numbers and the formulas just use add, subtract and multiply (which is why it is called linear programming).

### 3.10 THE INITIAL TABLEAU

In order to apply the simplex method to a maximum problem required the conversion to equations by using slack variables.

(i) Convert the inequalities to equations by adding a slack variable to the left side to give the slack equations:

$$\begin{cases} ax_1 + bx_2 \leq E \\ cx_1 + dx_2 \leq F \end{cases}$$

The slack equations are

$$\begin{cases} ax_1 + bx_2 + s_1 = E \\ cx_1 + dx_2 + s_2 = F \end{cases}$$

where s1 and s2 are the slack variables and are nonnegative and a, b, c, d, E and F denote any real number.



(ii) We are now ready to set up the matrix which represents the initial simplex tableau.

(a). the objective row is always the bottom row

(b). the slack equations form all the other rows

(c). the symbol for each variable appears above the column where its coefficients appear.

(d). the notation BV stands for basic variables. These are the variables that have only 0's and

1's in the column. The others are called non-basic variables.

(e). the notation RHS stands for the right-hand side of the equal sign in the slack equations.

## 3.11 DATA MODELING

Several algorithms are available to solve knapsack problems, based on dynamic programming approach, branch and bound approach or hybridizations of both approaches.

### 3.11.1 DYNAMIC PROGRAMMING

In terms of mathematical optimization, dynamic programming usually refers to simplifying a decision by breaking it down into a sequence of decision steps over time. This is done by defining a sequence of value functions $V_1, V_2, ..., V_n$, with an argument y representing the state of the system at times i from 1 to n. The definition of $V_n(y)$ is the value obtained in state y at the last time n. The values $V_i$ at earlier times i = n −1, n − 2, ..., 2, 1 can be found by working backwards, using a recursive relationship called the Bellman equation. For i = 2, ..., n, $V_{i-1}$ at any state y is calculated from $V_i$ by maximizing a simple function (usually the sum) of the gain from decision i − 1 and the function $V_i$ at the new state of the system if this decision is made. Since $V_i$ has already been calculated for the needed states, the above operation yields $V_{i-1}$ for those states. Finally, $V_1$ at the initial state of the system is the value of the optimal solution. The optimal values of the decision variables can be recovered, one by one, by tracking back the calculations already performed.

### 3.11.2 KNAPSACK PROBLEM

The knapsack problem is one of the most studied problems in combinatorial optimization, with many real-life applications. For this reason, many special cases and generalizations have been examined.

Common to all versions are a set of $n$ items, with each item $1 \leq j \leq n$ having an associated profit $p_j$ and weight $w_j$. The objective is to pick some of the items, with maximal total profit, while obeying that the maximum total weight of the chosen items

must not exceed $W$. Generally, these coefficients are scaled to become integers, and they are almost always assumed to be positive.

The knapsack problem in its most basic form:

$$maximize \sum_{j=1}^{n} p_j x_j$$

$$subject\ to \sum_{j=1}^{n} w_j x_j \leq W, \quad x_j \in \{0,1\} \forall j \in \{1, \ldots, n\}$$



Knapsack problems appear in real-world decision-making processes in a wide variety of fields, such as finding the least wasteful way to cut raw materials, selection of capital investments and financial portfolios, selection of assets for asset-backed securitization, and generating keys for the Merkle–Hellman knapsack cryptosystem.

One early application of knapsack algorithms was in the construction and scoring of tests in which the test-takers have a choice as to which questions they answer. For small examples it is a fairly simple process to provide the test-takers with such a choice. For example, if an exam contains 12 questions each worth 10 points, the test-taker need only answer 10 questions to achieve a maximum possible score of 100 points. However, on tests with a heterogeneous distribution of point values, that is, different questions are worth different point values, it is more difficult to provide choices. Feuerman and Weiss proposed a system in which students are given a heterogeneous test with a total of 125 possible points. The students are asked to answer all of the questions to the best of their abilities. Of the possible subsets of problems whose total point values add up to 100, a

knapsack algorithm would determine which subset gives each student the highest possible score.

## 3.12 TYPES OF KNAPSACK PROBLEM

### 3.12.1 MULTI-OBJECTIVE KNAPSACK PROBLEM



This variation changes the goal of the individual filling the knapsack. Instead of one objective, such as maximizing the monetary profit, the objective could have several dimensions. For example there could be environmental or social concerns as well as economic goals. Problems frequently addressed include portfolio and transportation logistics optimizations

There are many heuristics for solving this variant, including the Ant Colony Optimization algorithm.

As a concrete example, suppose you ran a cruise ship. You have to decide how many famous comedians to hire. This boat can handle more than one ton of passengers and the entertainers must weigh less than $1000\ lbs$. Each comedian has a weight, brings in business based on their popularity and asks for a specific salary. In this example you have multiple objectives. You want, of course, to maximize the popularity of your entertainers while minimizing their salaries. Also, you want to have as many entertainers as possible.

### 3.12.2 SUBSET-SUM KNAPSACK PROBLEM

The subset-sum problem is given a set of $n$ items and a knapsack, with

$w_j = weight\ of\ item\ j;$

$c = capacity\ of\ the\ knapsack,$

select a subset of the items whose total weight is closet to, without exceeding the total capacity of the Knapsack, $c$. That is;

$$maximize\ z = \sum_{j=1}^{n} w_j x_j$$

$$subject\ to\ \sum_{j=1}^{n} w_j x_j \leq c$$

$$x_j = 0 \text{ or } 1, \qquad j \in N = \{1, \ldots\ldots\ldots\ldots\ldots\ldots, n\}$$

where $x_j = \begin{cases} 1 & if\ item\ j\ is\ selected \\ 0 & if\ item\ j\ is\ not\ selected \end{cases}$

### 3.12.3 THE CHANGE-MAKING PROBLEM

The change-making problem can be interpreted as that of a cashier having to assemble a given change, $c$, using the least number of coins of specified values $w_j$ $(j = 1, \ldots\ldots\ldots, n)$ in the case where, for each value, an unlimited number of coins is available. The change-making problem can be viewed as an unbounded knapsack problem in which $p_j = -1$ for all $j$ and, in the capacity constraint, strict equality is imposed.

### 3.12.4 CUTTING-STOCK PROBLEM

The cutting-stock problem is an optimization problem, or more specifically, an integer linear programming problem. It arises from many applications in industry.

The standard formulation for the cutting-stock problem (but not the only one) starts with a list of m orders, each requiring $q_j$, j = 1,...,m pieces. A list of all possible combinations of cuts (often called "patterns"), associating with each pattern a positive integer variable $x_i$ representing how many times each pattern is to be used is constructed. The linear integer program is then:

Maximize $\sum_{i=1}^{n} c_i x_i$

Subject to $\sum_{i=1}^{n} a_{ij} x_i \geq q_j \qquad \forall j = 1, \dots, m$

and $x_i \geq 0, integer$

where $a_{ij}$ is the number of times order j appears in pattern i and $c_i$ is the cost (often the waste) of pattern i. The precise nature of the quantity constraints can lead to subtly different mathematical characteristics. The above formulation's quantity constraints are minimum constraints (at least the given amount of each order must be produced, but possibly more). When $c_i$=1 the objective minimises the number of utilised master items and, if the constraint for the quantity to be produced is replaced by equality, it is called the bin packing problem. The most general formulation has two-sided constraints (and in this case a minimum-waste solution may consume more than the minimum number of master items):

$q_j \leq \sum_{i=1}^{n} a_{ij} x_i \leq Q_j, \qquad \forall j = 1, \dots m$

This formulation applies not just to one-dimensional problems. Many variations are possible, including one where the objective is not to minimise the waste, but to maximise the total value of the produced items, allowing each order to have a different value.

In general, the number of possible patterns grows exponentially as a function of m, the number of orders. As the number of orders increases, it may therefore become impractical to enumerate the possible cutting patterns.



### 3.12.5 MULTIPLE KNAPSACK PROBLEMS

This variation is similar to the Bin Packing Problem. It differs from the Bin Packing Problem that a subset of items can be selected, whereas, in the Bin Packing Problem, all items have to be packed to certain bins. The concept is that there are multiple knapsacks. This may seem a trivial change, but it is not equivalent to adding to the capacity of the initial knapsack. This variation is used in many loading and scheduling problems in Operations Research and has a Polynomial Time Approximation Scheme (PTAS).

### 3.12.6 THE QUADRATIC KNAPSACK PROBLEM

The binary quadratic knapsack problem (QKP) is formally defined as follows: assume that n items are given where item j has a positive integer weight $w_j$. In addition, an $n \times n$ nonnegative integer matrix $P = \{p_{ij}\}$, where $p_{jj}$ is the profit achieved if item j is selected and $p_{ij} + p_{ji}$ is a profit achieved if both items i and j are selected for $i < j$ is given. The Quadratic Knapsack Problem calls for selecting an item subset

whose overall weight does not exceed a given knapsack capacity c, so as to maximize the overall profit. For notational convenience, let N := { 1 , … , n } denote the item set. By introducing a binary variable x j to indicate whether item j is selected, the problem may be formulated:

$$maximize \sum_{i \in N} \sum_{j \in N} p_{ij}\, x_i x_j$$

$$subject\ to\ \sum_{j \in N} w_j x_j \leq c \leq, \qquad x_j \in \{0, 1\},\ j \in N$$



Without loss of generality it is assumed that $max_{j \in N}\, w_j \leq c < \sum j \in N^{wj}$ and that the profit matrix is symmetric,(that is,$pij = pji$ for all $j > i$.

### 3.12.7 UNBOUNDED KNAPSACK PROBLEM

The unbounded knapsack problem (UKP) places no upper bound on the number of copies of each kind of item and can be formulated as

$$maximize \sum_{i=1}^{n} v_i x_i$$

$$subject\ to \sum_{i=1}^{n} w_i x_i \leq W, \qquad x_i \in \{0, 1, \ldots., c_i\}$$

except for that the only restriction on $\boldsymbol{x_i}$ is that it is a non-negative integer.

If all weights $(w_1, \ldots\ldots, w_n, W)$ are nonnegative integers, the knapsack problem can be solved in pseudo-polynomial time using dynamic programming. The following describes a dynamic programming solution for the unbounded knapsack problem.

To simplify things, assume all weights are strictly positive ($w_i > 0$). The objective is to maximize total value subject to the constraint that total weight is less than or equal to $W$. Then for each $w \leq W$, define $m[w]$ to be the maximum value that can be attained with total weight less than or equal to $w$. Then $m[W]$ is the solution to the problem.

Observe that $m[w]$ has the following properties:

$(i) m[0] = 0$, which is the sum of zero items and it is the summation of the empty set.

$(ii) m[w] = \max_{w_i \leq w}(v_i + m[w - w_i])$, where $v_i$ is the value of the $i - th$ kind of item.

Here the maximum of the empty set is taken to be zero. Tabulating the results from $m[0]$ up through $m[W]$ gives the solution. Since the calculation of each $m[w]$ involves examining $n$ items, and there are $W$ values of $m[w]$ to calculate, the running time of the dynamic programming solution is $O(nW)$. Dividing $w_1, w_2, \ldots\ldots, w_n, W$ by their greatest common divisoris an obvious way to improve the running time.

The $O(nW)$ complexity does not contradict the fact that the knapsack problem is NP-complete, since $W$, unlike $n$, is not polynomial in the length of the input to the problem. The length of the $W$ input to the problem is proportional to the number of bits in $W$, $\log W$, not to $W$ itself.

### 3.12.8 BOUNDED KNAPSACK PROBLEM

The Bounded Knapsack Problem removes the restriction that there is only one of each item, but restricts the number $x_i$ of copies of each kind of item to an integer value $c_i$.

Mathematically the bounded knapsack problem can be formulated as:

$$maximize \sum_{i=1}^{n} v_i x_i$$

$$subject\ to \sum_{i=1}^{n} w_i x_i \leq W\ , x_i \in \{0, 1, \dots, c_i\}$$



### 3.12.9 0-1 KNAPSACK PROBLEM

The most common problem being solved is the 0-1 knapsack problem, which restricts the number $x_i$ of copies of each kind of item to zero or one.

Mathematically the 0-1-knapsack problem can be formulated as:

Let there be $n$ items, $x_1$ to $x_n$ where $x_i$ has a value $v_i$ and weight $w_i$. The maximum weight that the bag can carry is $W$. It is common to assume that all values and weights are nonnegative. To simplify the representation, it is assume that the items are listed in increasing order of weight.

$$maximize \sum_{i=1}^{n} v_i x_i$$

$$subject\ to \sum_{i=1}^{n} w_i x_i \leq W,\ x_i \in \{0, 1\}$$

Maximize the sum of the values of the items in the knapsack so that the sum of the weights must be less than the knapsack's capacity.

A similar dynamic programming solution for the 0-1 knapsack problem also runs in pseudo-polynomial time. Assume $w_1, w_2, \ldots, w_n, W$ are strictly positive integers. Define $m[i, w]$ to be the maximum value that can be attained with weight less than or equal to $w$ using items up to $i$.



Thus $m[i, w]$ can be defined recursively as follows:

$(i) m[i, w] = m[i-1, w]$ if $w_i > w$ (the new item is greater than the existing weight limit).

$(ii) m[i, w] = \max(m[i-1, w], m[i-1, w-w_i] + v_i)$ if $w_i \leq w$.

The solution can then be found by calculating $m[n, W]$. To do this efficiently we can use a table to store preceding computations.

### 3.13 GENETIC ALGORITHM

A genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

Genetic algorithms find application in bioinformatics, phylogenetics, computational science, engineering, economics, chemistry, manufacturing, mathematics, physics, pharmacometrics and other fields.

In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and is an iterative process, with the population in each iteration called a generation. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

A typical genetic algorithm requires:

(i) a genetic representation of the solution domain,

(ii) a fitness function to evaluate the solution domain.

A standard representation of each candidate solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property

that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used, but crossover implementation is more complex in this case. Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary programming; a mix of both linear chromosomes and trees is explored in gene expression programming.



## 3.14 DUALITY AND INTEGER PROGRAMMING

### 3.14.1 DUALITY

Subsequent to any given linear programming problem, called the Primal Problem, there is another linear programming problem called the Dual Problem. Since a given linear programming problem can be stated in several forms (standard form, canonical form, general form etc), it follows that the form of the dual problem will depend on the form of the primal problem.

Dual's General LP i.e. converting a primal to a dual or the reverse.

(i). Multiply the objective function by -1 and change \max" to \min" or \min" to \max".

(ii). Multiply an inequality constraint by -1 to change the direction of the inequality.

(iii). Replace an equality constraint

$$\sum_{j=1}^{n} a_{ij} x_j = b$$

with two inequality constraints

$\sum_{j=1}^{n} a_{ij} x_j \leq b_i$

$$-\sum_{j=1}^{n} a_{ij}\, x_j = -b$$

(iv). Replace a variable that is non-positive with a variable that is its negative.

For example, if $x_j$ is specified to be non-positive by $x_j \leq 0$ , replace every occurrence of $x_j$ with $-\hat{x}j$ and require$-\hat{x}j \geq 0$

(v). Replace a variable that is unrestricted in sign with the difference of two non-negative variables. For example, if $x_j$ is unrestricted (sometimes called free), replace every occurrence of $x_j$ with $x_j^+ - x_j^-$ require that $x_j^+ - x_j^-$ be nonnegative variables.

Using these transformations, every Linear Programming (LP) can be converted into an equivalent one in standard form. In this case equivalent means that an optimal solution to the original problem can be obtained from an optimal solution to the new problem.

### 3.14.2 INTEGER PROGRAMMING

Integer Linear Program is a linear program with the additional requirement that some or all of the decision variables must be integer. An integer linear program is said to be an all-integer linear program if all of the variables are required to be integer.

When the phrase "integer" is dropped from a model, it will be left with a two-variable linear program. The linear program that results from dropping the integer requirements for decision variables is referred to as LP Relaxation of the Integer Linear Program.

An integer linear programming in which some but not all of the decision variables are required to be integer is called a mixed-integer program. When some or all the integer variables are only permitted to assume the values zero and one, then we have binary or

0-1 integer linear program. Capital budgeting and bank location problems are applications of 0-1 integer linear program.

## 3.15 METHODS FOR SOLVING LINEAR INTEGER PROGRAMMING OPTIMIZATION (LIP) PROBLEMS

The advance of precise optimization methods for LIP optimization problems during the last five decades had been very triumphant. At least, there are three different approaches for solving integer programming problems, although they are commonly combined into "hybrid" solution procedures in computational practice, which are considered briery as follows.

(i) Cutting planes algorithms based on polyhedral combinatorics.

(ii) Enumerative approaches and Branch and Bound, Branch and Cut and Branch and Price methods.

(iii) Relaxation and decomposition techniques.

Cutting Plane algorithms based on polyhedral combinatorics.

The fundamental idea of polyhedral combinatorics is to replace the constraint set of an integer programming problem by an alternative convexification of the feasible points and extreme rays of the problem. Both the size and the complexity of the problems solved have been increased considerably when polyhedral theory was applied to numerical problem solving.

The general cutting plane approach relaxes initially the integrality restrictions on the variables and solves the resulting linear program over the constraint system.

In case the linear program is unbounded or infeasible, the same is suitable for the integer program. In case the solution to the linear program is integer, this is the optimal solution

to the integer program. When the linear program has a not integer optimal solution, then a facet-identification problem has to be solved.

Here the objective is to find a linear inequality that "cuts off" the fractional linear programming solution while assuring that all feasible integer points satisfy the inequality, that is, an inequality that "separates" the fractional point from the polyhedron. The terminating conditions for this algorithm are as follows;

(i). an integer solution is found (the LIP problem is successfully solved).

(ii). the linear program is infeasible and therefore the integer problem is infeasible.

(iii). No cut is identified by the facet-identification procedures either because a full description of the facial structure is not known, or because the facet-identification procedures are inexact, that is, there is no possibility for algorithmically generating cuts of a known form.

In case the cutting plane procedure is terminated because of the third possibility, then, in general, the search process has "tightened" the linear programming formulation so that the resulting linear programming solution value is much closer to the integer solution value. Another strategy for cutting-plane algorithms is to maintain integrality and dual feasibility and then to use cuts to obtain primal feasibility.

**Enumerative approaches**

These approaches are known under different names. The most popular of them are Branch and Bound, implicit enumeration and divide and conquer. The explicit enumeration is the simplest approach to solving a pure integer programming problem by means of enumeration of all possibilities, which are finite in number.

However, due to the "combinatorial explosion" of number of these possibilities resulting from the parameter "size", only instances having relative small size could be solved by

such an approach within a reasonable computational time limit. Sometimes many possibilities can be implicitly eliminated by domination or feasibility arguments. Besides straight forward or implicit enumeration, the most commonly used enumerative approach is called Branch and Bound (B&B), where the "branching" refers to the enumeration part of the solution technique and "bounding" refers to the fathoming of possible solutions by comparison to a known upper or lower bound on the solution value. A variety of strategies that have been used within the general Branch and Bound framework are being described as follows;



**Branch and Cut**

The bounds obtained from the LP-relaxations are often weak, which may cause standard B&B algorithms to fail in practice. It is therefore of crucial importance to tighten the formulation of the problem to be solved. The idea of dynamically adding the cutting planes to the problem is one way of obtaining stronger bounds. Combining cutting plane algorithm with B&B results in a very powerful class of Branch and Cut (B&C) algorithms. The idea is to generate cutting planes throughout the B&B tree of a standard B&B algorithm, in order to get tight bounds at each node. The B&C algorithm consists of following major components:

(a). Automatic reformulation procedures.

(b). Heuristics which provide "good" feasible integer solutions.

(c). Cutting plane procedures which tighten the linear programming relaxation to the linear integer problem under consideration.

These components are embedded into a tree-search framework as in the B&B approach to integer programming; whenever possible, there is used a fourth component:

(d). the procedure permanently fixes variables (by reduced cost implications and logical implications) and does comparable conditional fixing throughout the search-tree.

These four components are combined so as to guarantee optimality of the solution obtained at the end of the calculation.

The increasing empirical evidence indicates that both pure and mixed integer programming problems can be solved to proven optimality in economically feasible computation times by methods based on the polyhedral structure of integer programs.



**Branch and Price**

The philosophy of Branch and Price (B&P) is similar to the one of Branch and Cut. Indeed, the pricing and the cutting are procedures for tightening the LP relaxation of the problem. In Branch and Price, the concept of column generation is combined with a Branch and Bound algorithm. The simplex algorithm arises at the origin from the column generation concept, where only variables with negative reduced costs are allowed to enter the basis at each iteration. Given an LP model with a huge number of variables, possibly depending exponentially on the instance size, it would be efficient to consider only the variables potentially improving the objective function. The main idea of column generation is to efficiently determine a variable with negative reduced costs to enter the basis, add it to the problem, resolve it and iteratively repeat this process until no variable with negative reduced costs exists any more.

In general, the method is often used for obtaining LP/LIP models with an exponential number of variables, which provide tighter bounds than the original compact LP/LIP pair. Since column generation is an algorithm for solving LPs, it has to be combined with another method in order to solve LIPs to optimality.

The B&P algorithm is the result of combining column generation with B&B problems. Routing and scheduling are the most suitable areas for application of Branch and Price methods.

From a theoretical point of view, B&C and B&P are closely related, since column generation in the primal problem corresponds to cut generation in the dual and vice-versa. Furthermore, B&C and B&P can be combined in the so called Branch and Cut and Price algorithms, where both cuts and variables are dynamically generated.



**Relaxation and Decomposition Method**

There are three wide spread approaches for relaxation of the general LIP problem, which are designed to find an upper bound of the optimal value for the maximizing LIP problem: Linear Programming (LP) relaxation, Combinatorial relaxation and Lagrangian relaxation. The first two approaches extend the feasible domain without any change in the objective function of the problem. The third approach provides another maximizing objective function, which has the same or greater value in a fixed feasible domain.

The LP relaxation for the Integer Programming model is obtained by dropping the integrality constraints on the variables. For realization of the combinatorial relaxation there are at least two approaches exploiting the combinatorial structure of the problem. The first approach is based on the concept of valuated matroids, introduced by Dress and Wenzel. The other approach, which is called the structural approach, utilizes algorithms to compute an upper bound on the objective function and is often based on a graph-theoretic method.

Considering LP relaxation, it was mentioned that relaxing the integrality restriction is one approach to solution of linear integer programming problems. But, this is not the

only approach to relaxing the problem. The idea of dropping constraints can be embedded into a more general framework, called Lagrangian relaxation. This is an alternative approach, where a set of "complicating" constraints is included into the objective function in a Lagrangian fashion (with fixed multipliers that are iteratively changed). The complicating constraints are removed from the constraint set. In this way the resulting sub-problem could be solved considerably easier.

To realize a Lagrangian relaxation it is necessary that the structure of the problem being solved is clear in order to relax then the constraints that are "complicating".

A related approach which attempts to strengthen the bounds of Lagrangian relaxation is called Lagrangian decomposition. This approach consists of isolating sets of constraints. In this way are obtained separately, easy problems to solve over each of the subsets. The dimension of the problem is increased by creating linking variables which link the subsets. All Lagrangian approaches are problem dependent. There is developed no general theory applicable to say, in arbitrary zero-one or LIP problems underlying polyhedral structure of these problems. Thus, in order to use this approach, one must be able to both identify specific mathematical structures inherent in the problem and then study the polyhedron associated with that structure.



### 3.16 BRANCH AND BOUND

The basic concept underlying the branch-and-bound technique is to divide and conquer. The process contains dividing (branching) original large problem into smaller sub problems and bounding the best solution in the subsets.

The steps are;

(i) Solve the problem without integer restrictions,

(ii) If the solution is integer , then this must be the solution to integer problem,

(iii) If these variables are not integer valued, the feasible region is divided by adding constraints restricting the value of one of the variables that was not integer valued,

(iv) Bounds on the value of the objective function are found and used to help determine which sub-problems can be eliminated and when the optimal solution has been found,

(v) If a solution is not optimal, a new sub-problem is selected and branching continues.

Branch and bound (BB or B&B) is a general algorithm for finding optimal solutions of various optimization problems, especially in discrete and combinatorial optimization. A Branch-and-Bound algorithm consists of a systematic enumeration of all admissible solutions, where large subsets of fruitless candidates are discarded en masse, by using upper and lower estimated bounds of the quantity being optimized.

### 3.17 GENERAL DESCRIPTION OF BRANCH AND BOUND

In order to facilitate a concrete description, it is assumed that the goal is to find the minimum value of a function $(x)$, where $x$ ranges over some set $S$ of admissible solutions (the search space or feasible region). The maximum value of $f(x)$ can be found by finding the minimum of $(x) = -f(x)$. For instance, $S$ could be

the set of all possible trip schedules for a plane flight, and $f(x)$ could be the expected revenue for schedule $x$.)

A Branch-and-Bound procedure requires two tools. The first one is a splitting procedure that, given a set $S$ of candidates, returns two or more smaller sets $S_1, S_2, \ldots$ whose union covers$S$. Make a note of the fact that, the minimum of $f(x)$ over $S$ is $\min\{v_1, v_2, \ldots\}$, where each $v_i$ is the minimum of $f(x)$ within $S_i$. This step is called branching, since its recursive application defines a tree structure (the search tree) whose nodes are the subsets of $S$.



The second tool is a procedure that computes upper and lower bounds for the minimum value of $f(x)$ within a given subset of $S$. This step is called bounding.

The key idea of the Branch and Bound algorithm is: if the lower bound for some tree node (set of candidates) $A$ is greater than the upper bound for some other node $B$, then $A$ may be safely discarded from the search. This step is called pruning, and is usually implemented by maintaining a global variable $m$ (shared among all nodes of the tree) that records the minimum upper bound seen among all sub regions examined so far. Any node whose lower bound is greater than $m$ can be discarded.

The recursion stops when the current candidate set $S$ is reduced to a single element, or when the upper bound for set $S$ matches the lower bound. Either way, any element of $S$ will be a minimum of the function within $S$.

For the purpose of this research, the Branch and Bound knapsack would be employed in solving the problem.

### 3.18 SENSITIVITY ANALYSIS

A technique used to determine how different values of an independent variable will impact a particular dependent variable under a given set of assumptions. This technique is used within specific boundaries that will depend on one or more input variables.

The Sensitivity analysis is intended to study the effect of changes in the parameters of the Linear Programming (LP) model on the optimal solution. Such analysis is regarded as an integral part of the (extended) solution of any LP problem. The sensitivity analysis gives the model a lively characteristic that allows the analyst to study the manners of the optimal solution as a result of making changes in the model's parameters. The decisive objective of the analysis is to obtain information about possible new optimum solutions (related to changes in the parameters) with minimal additional computations.

Sensitivity Problem 1: How much change is allowed in the objective function coefficients?

Changes in the objective function coefficients can affect only the slope of the straight line representing it. The optimum corner point of a given solution space depends totally on the slope of the objective function.

Sensitivity Problem 2: How much is the worth of a resource unit?

The problem deals with the study of the sensitivity of the optimum solution to changes in the right-hand side of the constraints. If the constraints represent a limited resource, the problem reduces to studying the effect of changing the accessibility of the resource. The precise goal of this sensitivity problem is to settle on the effect of changes in the right-hand side of constraints on the optimum objective value. In core, the results are

given as preset ranges of the right-hand side within which the objective optimum value will change (increase or decrease) at a given steady rate.



# CHAPTER FOUR

## DATA COLLECTION AND ANALYSIS

### 4.0 INTRODUCTION

The study area for this research is the Sekyere Central District Assembly. The Assembly is responsible for most development projects in the District. Budget is being proposed by the Assembly concerning projects within certain period of time.

Consideration is given to a computational lesson of the branch and Bound algorithm applied to Knapsack problem in this chapter.

Reflection is given to a 0-1 Knapsack Problem where $n \subset N$ such that $\sum_{i=1}^{n} \leq b$, where each item has a profit or cost $c_i$ and a weight $w_i$. The task is to choose a subset of items whose total weight does not exceed the Knapsack capacity $b$, and whose total profit is the maximum.

Generally, it assumed that all input data are positive integers. With the knowledge of the binary decision variable $x_i$ with $x_i = 1$, if item $i$ is selected, and $x_i = 0$, if not, an integer linear programming is obtained:

$$maximize\ Z = \sum_{i=1}^{n} c_i x_i$$

$subject\ to\ \sum_{i=1}^{n} w_i x_i \leq b$, where $x \in \{0,1\}^N, i \in Z^+$.

$w_i < b$ for $i \in Z^+$ may be assumed to make sure that each item measured fits into the Knapsack, and that $\sum_{i=1}^{n} c_i > b$ to avoid insignificant solution.

The Knapsack problem becomes applicable in real life situations at the Sekyere Central District Assembly in selecting of communities for a unit classroom building in the District. The Assembly intends to optimize the land capacity allocated for the Unit

classroom within the District so that the District gets the best usage of land at a minimal cost.

Several lands at different locations have been considered for the construction of the Unit classroom within the District but a suitable land needs to be developed to pave way for the construction of the Building. Though the lands have been allocated already but the Knapsack problem becomes useful in getting the optimal land for the construction of the Unit classroom block.



## 4.1 DATA COLLECTION AND ANALYSIS

The analysis will be centered on data available at the Budget office of the Sekyere Central District Assembly with Nsuta as the District capital in the Ashanti Region.

In an effort to develop the educational infrastructure in the District, the Assembly proposed a budget of $GH¢360{,}000.00$ for the development of lands and the construction of Unit classroom buildings. These buildings are to be constructed at ten different towns within the district, whose estimated capacities in terms of the number of unit classrooms and development cost are given in the table 4.1. The respective towns to be considered within the District are Kwaman, Nsuta, Beposo, Bonkuro, Kurowe, Appiah Kurom, Kyekyebon, Asasebonso, Jeduakoo and Atwea,

**Table 4.1: Respective towns with their Budget allocations**

| Towns | Capacity (unit classroom) | Cost (GH¢ 1000) |
|---|---|---|
| Kwaman | 3 | 74 |
| Nsuta | 6 | 150 |
| Beposo | 9 | 210 |
| Bonkuro | 3 | 62 |
| Kurowe | 3 | 92 |
| Appiah Kurom | 6 | 130 |
| Kyekyebon | 3 | 84 |
| Asasebonso | 6 | 125 |
| Jeduako | 3 | 68 |
| Atwea | 6 | 134 |

A unit classroom is made up of the number of study rooms, an office, a store, staff common room and a toilet facility. The type of unit classroom to be built in each town was based on the population, existing educational infrastructures. All three unit classrooms are for JHS, six unit classroom is for the lower and upper primary schools, and the nine unit classroom is for both primary and JHS. The difference in the cost for the same unit classroom was due to different construction works to be done on the various lands. Appendix 1 provides the breakdown of the budget of associated cost for each unit classroom for the various locations.

The dilemma here is to choose suitable locations in such a way that the optimal capacity would be attained without exceeding the budget allocated for project.

With a link to the Knapsack Problem model, the holding capacity of the resource maximum value is the Assembly's budget. The various items to be measured are the different sites (lands) that can be developed for the project, the weight of any item is the cost of developing and construction of the project and the value of each item is the capacity of each site.

The problem can therefore be modeled as:



$$Maximize\ C = \sum_{i=1}^{n} c_i x_i$$

$$Subject\ to \sum_{i=1}^{n} w_i x_i \leq W, \quad x_i \in \{0,1\}^N, \ \ i = 1, \ldots\ldots, n$$

Where;

$C =$ Total capacity

$c_i =$ Capacity of each item or site

$x_i =$ Number of sites developed

$w_i =$ Cost of developing a site

$W =$ Total budget for the development (resource limit)

Thus,

$$maximize\ C = 3X_1 + 6X_2 + 9X_3 + 3X_4 + 3X_5 + 6X_6 + 3X_7 + 6X_8 + 3X_9 + 6X_{10}$$

$$subject\ to\ 74X_1 + 150X_2 + 210X_3 + 62X_4 + 92X_5 + 130X_6 + 84X_7 + 125X_8 + 68X_9 + 134X_{10} \leq 360$$

A Branch and Bound algorithm model is applied to carry out the computation of the model. The items to be considered are seven (which means $n = 10$) consisting of Kwaman, Nsuta, Beposo, Bonkuro, Kurowe, Appiah Kurom, Kyekyebon, Asasebonso, Jeduakoo and Atwea.

The weights of each item are $w_1 = 74, w_2 = 150, w_3 = 210, w_4 = 62, w_5 = 92, w_6 = 130, w_7 = 84, w_8 = 125, w_9 = 68, w_{10} = 134$ whiles the values of each item are $X_1 = 3, X_2 = 6, X_3 = 9, X_4 = 3, X_5 = 3, X_6 = 6, X_7 = 3, X_8 = 6, X_9 = 3, X_{10} = 6$ and the maximum available budget fund $W = 360$.

**Note: For locations**

| | | | |
|---|---|---|---|
| X1: | Kwaman | X6: | Appiahkurom |
| X2: | Nsuta | X7: | Kyekyebon |
| X3: | Beposo | X8: | Asasebonso |
| X4: | Bonkuro | X9: | Jeduako |
| X5: | Kurowe | X10: | Atwea |



## 4.2 RESULTS OF THE ANALYSIS

Results of the analysis in obtaining maximum number of unit classroom buildings at selected location in the district are shown in the tables below. The tables provide a breakdown of the associated cost in building the unit classroom. The optimal selection of unit classrooms yielded three hundred and seventeen thousand Ghana Cedis (GH¢317,000). The amount is able to construct a one 3-unit classroom building at Bonkuro and two 6-unit classroom building at Appiah Kurom and asasebonso respectively. Thus the total number of classroom to be built out of budget is 15. This means that out of the total budget of Three Hundred and Sixty Thousand Ghana Cedis (GH¢ 360,000) which was proposed by the Assembly, an excess amount of Forty Three Thousand Ghana Cedis (GH¢ 43,000) was left. This excess amount can be used to undertake other project in the District. Different budget is allocated for the construction

of a 6-unit classroom at appiahkurom and Asasebonso owing to their site location and kind of construction works to undertake. The budget allocated for the different locations that was not selected can be found in the Appendices.



**Tables for selected locations for the construction of unit classrooms**

**Table 4.2: Breakdown of Budget allocation for 3-unit classroom at Bonkuro**

| Item | Description<br>Bonkuro | Amount<br>GH¢ |
|---|---|---|
| | Construction of 1 no. 3 unit classroom with office, store, toilet facility and staff common room | |
| A | Preliminaries | 1,250 |
| B | Excavation and Earthworks | 19,382 |
| C | Concrete Works | 5,362 |
| D | Block works | 3,016 |
| E | Roofing to summary | 2,383 |
| F | Carpentry Works | 8,235 |
| G | Joinery/Walling | 293 |
| H | Metal works | 7,215 |
| I | Plastering works /floor | 3,142 |
| J | Painting/decoration | 2,900 |
| K | External works | 4,212 |
| L | Construction of ramps | 150 |
| M | Electrical works | 3,189 |
| N | Additional Amount | 1,271 |
| | **Total** | **62,000** |

The maximum amount required constructing a 3-unit classroom building at Bonkuro is sixty two thousand Ghana Cedis. This is the amount allocated to construct the 3-unit classroom building with an office, a store, staff common room and toilet facilities. Required amount needed to acquire various items in the construction of the 3-unit classroom have been shown in table 4.2 above.



**Table 4.3: Breakdown of Budget allocation for 6-unit classroom at Appiahkurom**

| Item | Description<br>Appiah Kurom | Amount<br>GH¢ |
|---|---|---|
| | Construction of 1 no. 6 unit classroom with office, store, toilet facility and staff common room | |
| A | Preliminaries | 1,045 |
| B | Excavation and Earthworks | 26,321 |
| C | Concrete Works | 8,123 |
| D | Block works | 5,179 |
| E | Roofing to summary | 6,219 |
| F | Carpentry Works | 6,321 |
| G | Joinery/Walling | 16,091 |
| H | Metal works | 4,892 |
| I | Plastering works /floor | 13,010 |
| J | Painting/decoration | 3,952 |
| K | External works | 3,289 |
| L | Construction of ramps | 289 |
| M | Electrical works | 4,358 |
| N | Additional Amount | 30,911 |
| | **Total** | **130,000** |

The maximum amount required constructing a 6-unit classroom building at Appiahkurom is One Hundred and Thirty Thousand Ghana Cedis (GH¢ 130,000). This amount allocated in constructing the 6-unit classroom building with an office, a store, staff common room and toilet facilities. Required amount needed to purchase various items in the construction of the 6-unit classroom have been shown in table 4.3 above.

**Table 4.4: Breakdown of Budget allocation for 6-unit classroom at Asasebonso**

| Item | Description<br>Asasebonso | Amount<br>GH¢ |
|---|---|---|
| | Construction of 1 no. 6 unit classroom with office, store, toilet facility and staff common room | |
| A | Preliminaries | 2,089 |
| B | Excavation and Earthworks | 35,105 |
| C | Concrete Works | 11,254 |
| D | Block works | 7,925 |
| E | Roofing to summary | 8,710 |
| F | Carpentry Works | 8,590 |
| G | Joinery/Walling | 18,127 |
| H | Metal works | 510 |
| I | Plastering works /floor | 14,328 |
| J | Painting/decoration | 5,610 |
| K | External works | 6,000 |
| L | Construction of ramps | 345 |
| M | Electrical works | 4,912 |
| N | Additional Amount | 1,495 |
| | **Total** | **125,000** |

The maximum amount required constructing a 6-unit classroom building at Asasebonso is One Hundred and Twenty Five Thousand Ghana Cedis (GH¢ 125,000). This is the amount allocated to construct the 6-unit classroom building with an office, a store, staff common room and toilet facilities by the Assembly. Required amount needed to acquire various items in the construction of the 3-unit classroom have been shown in table 4.2 above.



## 4.3 SENSITIVITY ANALYSIS ON THE WHOLE SOLUTION

The Sensitivity Analysis is often used for integer linear programming problem than the Linear Programming (LP) problem. That is, a very small change in one of the coefficients in the constraints can cause a reasonably large change in the optimal value. In the case of our study, any time there is a change in any of the amount of the budget allocation, and then the integer linear program problem has to be resolved with slight variation in the coefficients before an optimal solution is chosen for implementation.

# CHAPTER FIVE

## CONCLUSIONS AND RECOMMENDATIONS

### 5.0 INTRODUCTION

The selection of sites in construction of unit classroom buildings has been described as a 0-1 knapsack problem. Considering a 0-1 knapsack problem as an NP hard, a Branch and Bound method was used to solve the problem of selecting sites for construction of unit classroom.



### 5.1 CONCLUSION

The research sought to use the Knapsack problem for selecting required sites in critical situations such as construction of school buildings. However, it can be applied to any situation where allocation of funds in the sector of development becomes a serious problem. A minimum amount of three hundred and seventeen thousand Ghana cedis(GH¢317,000) was obtained in construction of a one 3-unit and two 6-unit classroom buildings at three different locations within the district to enhance the educational development.

### 5.2 RECOMMENDATIONS

1. The use of the quantitative management software is apparent and methodical as compared to chance method and should be used in the allocation of funds.

2. Excess amount of (GH¢43,000) can be used by the Sekyere Central District Assembly to initiate different developmental project.

## REFERENCE


1. Aggarwal and Hartline (2006). Knapsack Auctions.www.research.microsoft.com
2. Alberto Caprara, David Pisinger and Paolo Toth(2007). Exact Solution of the Quadratic Knapsack Problem. Journals of Operations Research 55:1001-1021
3. Balasubramanian Ram and Sanjiv Sarin(1988). An Algorithm for the 0-1 Equality Knapsack Problem. Journal of the Operational Research Society 39, 1045–1049.
4. Bertsimas D., C. Darnell, and R. Soucy (1999). Portfolio construction through mixed-integer programming at Grantham, Mayo, Van Otterloo and Company. Interfaces 29, n1, Jan. – Feb. 1999, 49-66.
5. Boryczka. U (2006). The ihfluence of Trial representation in ACO for good results in MKP. From Proceeding (505) Advances in Computer Science and Technology
6. Chang Sung-Ho (1998). Tactical-Level Resource allocation procedure for the hotel industry. Journals of Texas A & M Industrial and Systems Engineering.
7. Chocolaad Christopher A. (1998). Solving Geometric Knapsack Problems using Tabu Search Heuristics. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier
8. Claessens T., N. Van Dijk, and P.J. Zwaneveld (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. Deniz Dizdar, Alex Gershkov,and Benny Moldovanu(2010). Revenue maximization in the dynamic knapsack problem. Theoretical Economics 6 (2011), 157–184
11. Der-Chyuan Lou and Chin-Chen Chang (1997). International Journal of High Speed Computing (IJHSC). Change in behavior of outputs generated on varying the crossover and mutation rates.
12. Dantzig. G. B, (1963) Linear Programming and Extensions, Princeton University Press, Princeton, NJ.
13. Doprado Marques Fabiano and Nereu Arenales Marcos (2007).The constrained compartmentalised knapsack problem. Journals of Computers and Operations research.
14. 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.
15. Eleni Hadjiconstantinou and Nicos Christofides(2010). An exact algorithm for general, orthogonal, two-dimensional knapsack problems. European Journal of Operational Research
16. Esther M. Arkin, Samir Khuller and Joseph S. B. Mitchell (1993). Geometric knapsack problems. http://www.springerlink.com/content/g007w81p153h3326
17. Garg. M. L and Sunanda Gupta (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

18. Geir Dahl (1997). An introduction to convexity, polyhedral theory and combinatorial optimization. University of Oslo, Department of Informatics.
19. **George S. Lueker (1995).** Average-Case Analysis of Off-Line and On-Line Knapsack Problems. Journal of Algorithms Volume 29, Issue 2, Pages 277-305
20. Ghoseiri K., F. Szidaroyszky, and M. J. Asgharpour (2004). A multi-objective train scheduling model and solution. Transportation research part B: Methodological 38, 927.
21. Granmo O. C., B. J. Oommen, S. A. Myrer, and M. G. Olsen (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.
22. Gutierrez and Maria Talia (2007). Lifting general integer programs. Kansas State University Masters thesis.
23. Haghani A. and Y. Shafali (2002). Bus maintenance systems and scheduling: model formulations and solutions. Transportation research part A: Policy and Practice, 36, 453.
24. Higgins A., E. Kozan, and L. Ferreira (1996). Optimal scheduling of train on a single line track. Transportation research part B: Methodological, 38, 927.
25. Hillier F. S. and G. J. Lieberman (2001). Introduction to Operations research. McGraw-Hill, New York 576-581.
26. Horowi E and Sahni (1974). Computing partitions with applications to knapsack problems. Journal of ACM21, 277-292.
27. Huschka Bryce (2007). Finding adjacent facet-defining inequalities. Kansas State University Masters thesis.

28. Islam Mohammad Tauhidul(2009). Approximation algorithms for minimum knapsack problem. Master's degree Thesis, university of lethbridge
29. Johnson. E, Mehrotra. A, Nemhauser. G (1993). Min-cut clustering Mathematics Programming.
30. Kalai, R.and Vanderpooten, D.(2006). Lexicographic α-Robust Knapsack Problem. http://ieeexplore.ieee.org/xpl/freeabs
31. Karp R. m. (1972). Reducibility among combinatorial problems. Complexity of computer computations; Plenum Press New York 85-103.
32. **Kosuch. S and Lisser. A** (2009). On two-stage stochastic knapsack problems. Discrete Applied Mathematics Volume 159, Issue 16
33. Lawler. E. L (1977). Fast approximation algorithms for knapsack problems. Focs, pp.206-213 18th Annual Symposium on Foundations of Computer Science.
34. Lei Shi, Zhanhong Wang, Yao Yao, Lin Wei(2010). Streaming Media Caching Model Based on Knapsack Problem. Journal of Networks, Vol 6, No 9 (2011), 1379-1386.
35. **Lü Xin and Feng Denggu(2004).** Quantum algorithm analysis of knapsack problem. JOURNAL OF BEIJING UNIVERSITY OF AERONAUTICS AND A, V 30(11)
36. Mattfeld D. C. and H. Kopfer (2003). Terminal operations management in vehicle transshipment. Transportation research part A: Policy and Practice, 37, 435
37. Maya Hristakeva and Dipti Shrestha(2011). Solving the 0-1 Knapsack Problem with Genetic Algorithms. http://freetechebooks.com/file-2011/knapsack-problem

38. Maya Hristakeva and Dipti Shrestha(2005). Different Approaches to Solve the 0/1 Knapsack Problem. http://micsymposium.org/mics_2005/papers/paper 102.
39. Nemhauser G.L. and L. A. Wolsey (1998). Integer and Combinatorial Optimization. John Wiley and Sons, New York.
40. Oppong Emmanuel Ofori(2009). Optimal resource Allocation Using Knapsack Problems: A case Study of Television Advertisements at GTV. Master's degree thesis, KNUST.
41. Rajeev Kohli and Ramesh Krishnamurti(1992). A total-value greedy heuristic for the integer knapsack problem. Operations Research Letters Volume 12, Issue
42. Renata Mansini and M Grazia Speranza (2009). An Exact Algorithm for the Multidimensional Knapsack Problem. http://ideas.repec.org/a/eee/ejores/v196y2009i3p909-918
43. Ronghua Shang, Wenping Ma and Wei Zhang (2006). Immune Clonal MO Algorithm for 0/1 Knapsack Problems. Lecture Notes in Computer Science, 2006, Volume 4221/2006, 870-878.
44. Stefanie Kosuch(2010). An Ant Colony Optimization Algorithm for theTwo-Stage Knapsack Problem. http://www.kosuch.eu/stefanie
45. Stefanie Kosuch, Marc Letournel and Abdel Lisser (2009). On a Stochastic Knapsack Problem. Laboratoire de recherche en Informatique, Universite Paris Sud 91405 Orsay Cedex.
46. Stefanie Kosuch, Pierre Le Bodic, Janny Leung and Abdel Lisser(2009). On Stochastic Bilevel Programming Problem with Knapsack Constraints. http://www.kosuch.eu/stefanie/veroeffentlichungen

47. Srisuwannapa, C. and P. Charnsethikul( 2007). An Exact Algorithm for the Unbounded Knapsack Problem with Minimizing Maximum Processing Time. Journal of Computer Science, 3: 138-143.
48. Shahadat Khan, Kin F. LI, Eric G Manning and M D Mostofa Akbar (2002).SOLVING the knapsack problem for adaptive multimedia systems. http://studia.complexica.net/Art/RI020108
49. 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.
50. **Ulrich Pferschy , David Pisinger, and Gerhard J. Woeginger(1995).** Simple but efficient approaches for the collapsing knapsack problem. Journals of Operations Research.
51. Vineet Goyal R. Ravi (2003). Chance Constrained Knapsack Problem with Random Item Sizes. http://www.columbia.edu/~vg2277/stoch_knapsack
52. 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.
53. Witzgall. C, (1975). Mathematical methods of site selection for electronic message system (EMS), Technical Report, NBS Internal Report.
54. Xuan. M. A (2009). Artificial fish swarm algorithm for multiple knapsack problem. Journal of Computer Applications 2010, 30(2) 469-471
55. Yan S. and H. L. Chen (2002). A scheduling model and a solution algorithm for inter-city bus carriers. Transportation research part A: Policy & Practice, 36, 805.

Michel, S Perrot N, and Vanderbeck F (2009). Knapsack problems with setups http://ieeexplore.ieee.org/xpl/freeabs

56. Yunhong Zhou and Victor Naroditskiy (2008). Algorithm for Stochastic MultipleChoice Knapsack Problem and Application to Keywords Bidding. http://research.yahoo.com/workshops/troa-2008/papers/submission



## APPENDICES

**Table A.1: Breakdown of Budget Allocations for Various Sites**

| Item | Description | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | X9 | X10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A | Preliminaries | 1,320 | 2,680 | 5,680 | 1,250 | 1,320 | 1,045 | 1,320 | 2,089 | 1320 | 2,680 |
| B | Excavation and Earthworks | 22,723 | 44,100 | 74,100 | 19,382 | 25,578 | 26,321 | 23,287 | 35,105 | 20,152 | 46,109 |
| C | Concrete works | 6,263 | 12,970 | 13,970 | 5,362 | 9,287 | 8,123 | 7,540 | 11,254 | 6,263 | 9,870 |
| D | Block works | 5,016 | 8,960 | 9,960 | 3,016 | 7,546 | 5,179 | 7,546 | 7,925 | 5,016 | 7,328 |
| E | Roofing to summary | 3,383 | 8,800 | 9,800 | 2,383 | 5,624 | 6,219 | 5,129 | 8,710 | 4,215 | 8,147 |
| F | Carpentry works | 4,679 | 8,590 | 9,590 | 8,235 | 5,124 | 6,321 | 5,124 | 8,590 | 4,679 | 7,325 |
| G | Joinery/Walling | 10,891 | 22,980 | 39,980 | 7,215 | 11,298 | 16,091 | 10,098 | 18,127 | 9,254 | 22,980 |
| H | Metal works | 275 | 619 | 780 | 293 | 450 | 4,892 | 379 | 510 | 275 | 5,210 |
| I | Plastering work/floor | 8,162 | 17,010 | 18,500 | 3,142 | 9,254 | 13,010 | 8,321 | 14,328 | 7,105 | 15,219 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| J | Painting/decoration | 3,692 | 5,610 | 6,958 | 2,900 | 5,454 | 3,952 | 4,958 | 5,610 | 3,201 | 4,017 |
| K | External works | 3,000 | 6,000 | 7,000 | 4,212 | 4,129 | 3,289 | 4,092 | 6,000 | 285 | 4,987 |
| L | Construction of ramps | 150 | 345 | 566 | 150 | 350 | 289 | 350 | 345 | 180 | 345 |
| M | Electrical works | 3,189 | 5,888 | 9,953 | 3,189 | 5,240 | 4,358 | 4,890 | 4,912 | 4,210 | 4,328 |
| N | Surplus Amount | 1,259 | 5,448 | 3,163 | 1,271 | 1,346 | 30,911 | 966 | 1,495 | 1,845 | 1,455 |
| | **Total** | **74,000** | **150,000** | **210,000** | **62,000** | **92,000** | **130,000** | **84,000** | **125,000** | **68,000** | **134,000** |

**Table A.2 Description of 3-unit classrooms**

| Description | Quantity |
|---|---|
| Number of classrooms | 3 |
| Staff common room | 1 |
| Office | 1 |
| Store | 1 |
| Toilet (4 seater) & Washroom facility | 1 |



**Table A.3: Description of 6-unit classrooms**

| Description | Quantity |
|---|---|
| Number of classrooms | 6 |
| Staff common room | 1 |
| Office | 1 |
| Store | 1 |
| Toilet (4 seater) & Washroom facility | 1 |

**Table A.4: Description of 9-unit classrooms**

| Description | Quantity |
|---|---|
| Number of classrooms | 9 |
| Staff common room | 1 |
| Office | 1 |
| Store | 1 |
| Toilet (6 seater) & Washroom facility | 1 |

**Table A.5: Optimal Solutions for the various iterative stages (output from QM software)**

| Iteration | Level | Added Constraint | Sol. Type | Sol. Value | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | X9 | X10 | Cost |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Optimal | 15 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 317 |
| 1 | 7 | X5<=0 | INTEGER | 15 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 317 |
| 2 | 8 | X6<=0 | Suboptimal | 12 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 279 |
| 3 | 9 | X8<=0 | Suboptimal | 12 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 284 |
| 4 | 10 | X4<=0 | INTEGER | 15 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 347 |
| 5 | 10 | X5<=0 | INTEGER | 15 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 339 |
| 6 | 8 | X5<=0 | INTEGER | 15 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 337 |
| 7 | 10 | X5<=0 | INTEGER | 15 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 359 |
| 8 | 9 | X5<=0 | INTEGER | 15 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 342 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 7 | X5<=0 | INTEGER | 15 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 345 |
| 10 | 8 | X7<=0 | INTEGER | 15 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 353 |
| 11 | 10 | X5<=0 | INTEGER | 15 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 349 |
| 12 | 8 | X5<=0 | INTEGER | 15 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 350 |
| 13 | 9 | X7<=0 | INTEGER | 15 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 358 |
| 14 | 10 | X5<=0 | INTEGER | 15 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 354 |
| 15 | 10 | X5<=0 | INTEGER | 15 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 329 |
| 16 | 8 | X5<=0 | INTEGER | 15 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 346 |
| 17 | 9 | X5<=0 | INTEGER | 15 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 356 |
| 18 | 6 | X2>=1 | INTEGER | 15 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 360 |
| 19 | 10 | X5<=0 | INTEGER | 15 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 340 |
| 20 | 10 | X5<= | INTEG | 15 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 33 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | ER | | | | | | | | | | | | 5 |
| 21 | 7 | X5<=0 | INTEGER | 15 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 329 |
| 22 | 8 | X1<=0 | INTEGER | 15 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 347 |
| 23 | 9 | X8<=0 | Suboptimal | 12 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 296 |
| 24 | 10 | X4<=0 | INTEGER | 15 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 359 |
| 25 | 8 | X5<=0 | INTEGER | 15 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 339 |
| 26 | 10 | X5<=0 | INTEGER | 15 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 351 |
| 27 | 8 | X5<=0 | INTEGER | 15 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 384 |
| 28 | 10 | X5<=0 | INTEGER | 15 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 343 |
| 29 | 9 | X5<=0 | INTEGER | 15 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 340 |
| 30 | 10 | X5<=0 | INTEGER | 15 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 352 |
| 31 | 8 | X5<=0 | INTEGER | 15 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 334 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 32 | 9 | X1<=0 | INTEGER | 15 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 352 |
| 33 | 9 | X5<=0 | INTEGER | 15 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 344 |
| 34 | 10 | X5<=0 | INTEGER | 15 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 356 |
| 35 | 10 | X5<=0 | INTEGER | 15 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 348 |
| 36 | 10 | X5<=0 | INTEGER | 15 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 323 |
| 37 | 8 | X5<=0 | INTEGER | 15 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 321 |
| 38 | 9 | X8<=0 | Suboptimal | 12 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 288 |
| 39 | 10 | X4<=0 | INTEGER | 15 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 351 |
| 40 | 10 | X5<=0 | INTEGER | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 343 |
| 41 | 9 | X5<=0 | INTEGER | 15 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 346 |
| 42 | 8 | X5<=0 | INTEGER | 15 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 354 |
| 43 | 10 | X5<= | INTEG | 15 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 35 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | ER | | | | | | | | | | | | 8 |
| 44 | 10 | X5<=0 | INTEGER | 15 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 333 |
| 45 | 10 | X5<=0 | INTEGER | 15 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 344 |
| 46 | 8 | X5<=0 | INTEGER | 15 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 338 |
| 47 | 9 | X1<=0 | INTEGER | 15 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 356 |
| 48 | 9 | X5<=0 | INTEGER | 15 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 348 |
| 49 | 8 | X1>=1 | INTEGER | 15 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 360 |
| 50 | 10 | X5<=0 | INTEGER | 15 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 352 |
| 51 | 10 | X5<=0 | INTEGER | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 327 |
| 52 | 9 | X5<=0 | INTEGER | 15 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 326 |
| 53 | 10 | X4<=0 | INTEGER | 15 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 356 |
| 54 | 10 | X5<=0 | INTEGER | 15 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 348 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 55 | 10 | X5<=0 | INTEGER | 15 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 338 |
| 56 | 10 | X5<=0 | INTEGER | 15 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 332 |