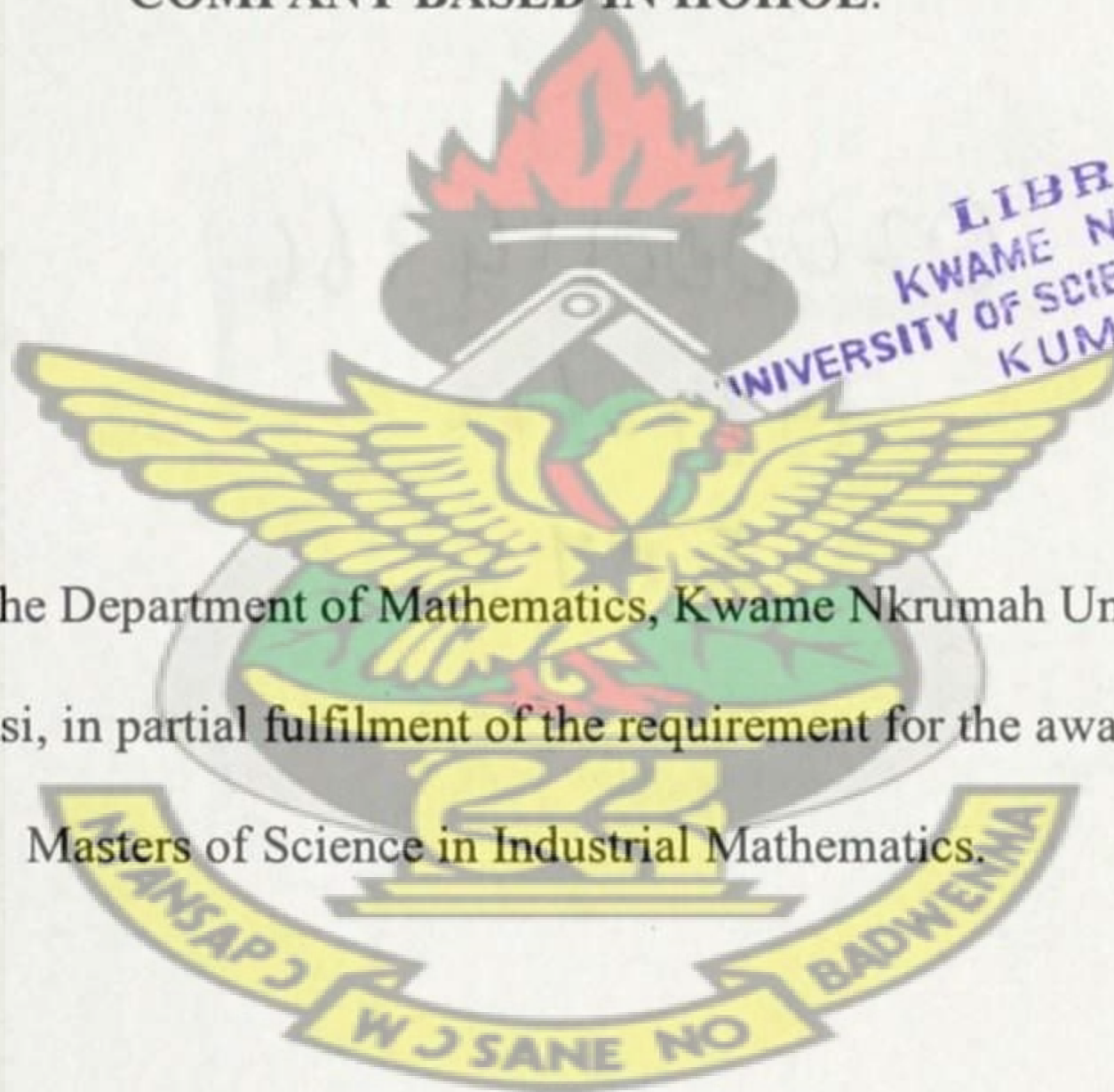


KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF MATHEMATICS

APPLICATION OF DYNAMIC PROGRAMMING:

**A CASE STUDY OF PRODUCT ALLOCATION PROBLEM OF A DISTRIBUTION
COMPANY BASED IN HOHOE.**



A Thesis Submitted to the Department of Mathematics, Kwame Nkrumah University of Science and Technology, Kumasi, in partial fulfilment of the requirement for the award of the degree of Masters of Science in Industrial Mathematics.

DZAMESI PROSPER DZIFA

June, 2012

DECLARATION

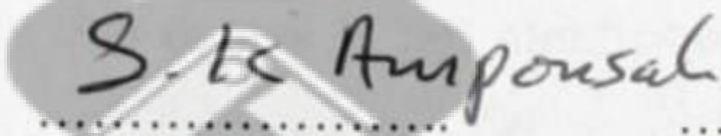
I hereby declare that this submission is my own work towards the Master of Science degree. And that, to the best of my knowledge, it contains no material(s) previously published by another person(s) nor material(s), which have been accepted for the award of any degree of the University, except where the acknowledgement has been made in the text.

Dzamesi Prosper Dzifa-(PG4065210)
(Student Name & ID No.)


Signature

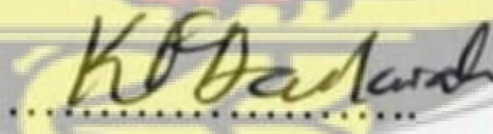
11-05-2012
Date

Certified by:
Dr. Samuel K. Amponsah
(Supervisor)

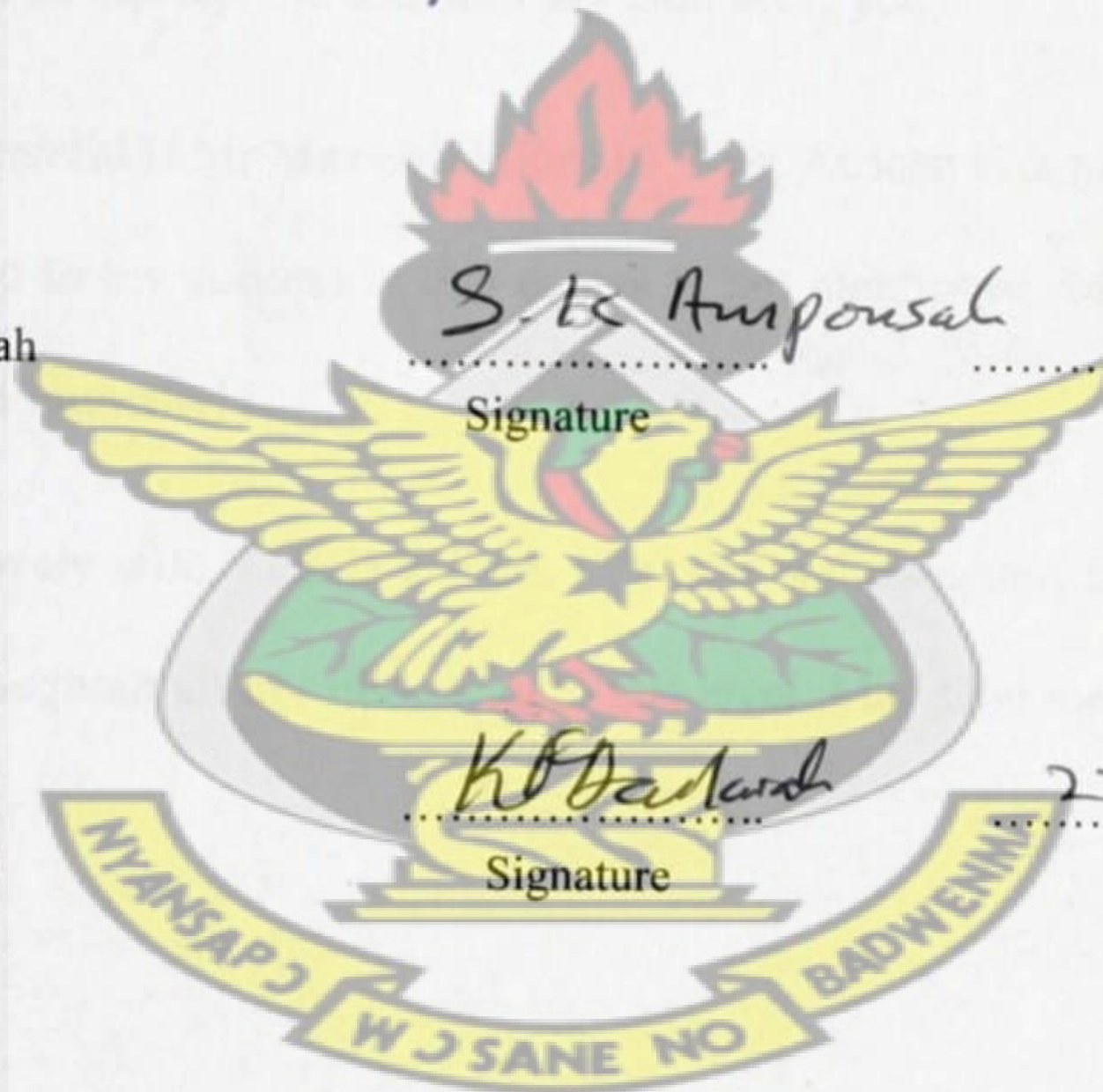

Signature

23/5/2012
Date

Certified by:
Mr. F.K. Darkwah
(Head of Department)


Signature

23/5/2012
Date



ACKNOWLEDGEMENT

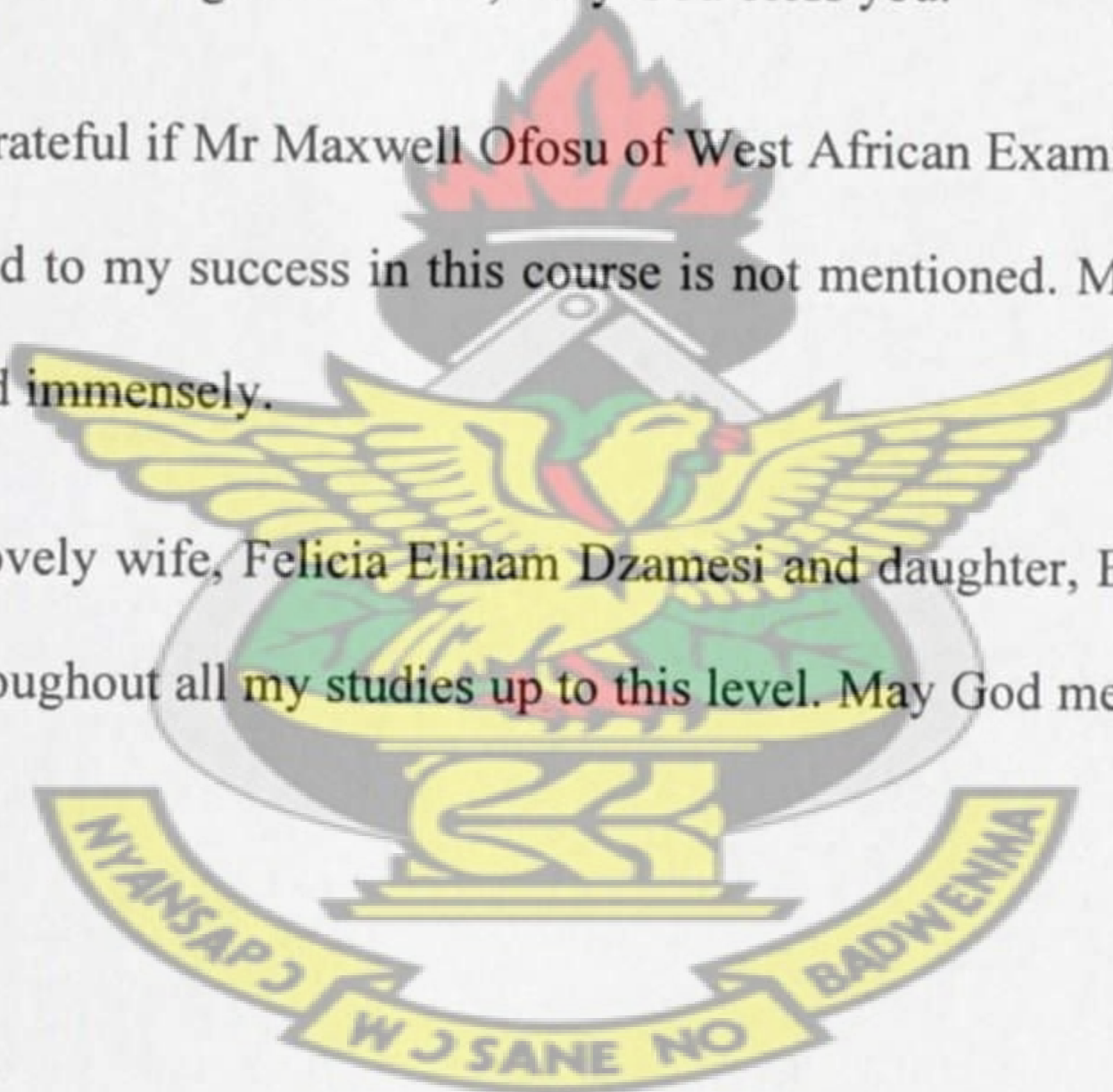
I am very grateful to the Almighty God for guiding me through the entire course and giving me success.

My sincerest gratitude goes to my supervisor, Dr S.K. Amponsah, who has supported me throughout my thesis with his patience and knowledge whilst allowing me the room to work under his supervision. I attribute the level of my Masters degree to his advanced knowledge and experience that he put to play in writing of this thesis.

My unquestionable appreciation also goes to all the other lecturers of Mathematics Department, who took us through the course, I say God bless you.

I will sound very ungrateful if Mr Maxwell Ofose of West African Examination Council who has greatly contributed to my success in this course is not mentioned. May God richly bless him and his household immensely.

Finally, I thank my lovely wife, Felicia Elinam Dzamesi and daughter, Enam Ama Dzamesi for supporting me throughout all my studies up to this level. May God meet you at your point of need. Stay blessed.



DEDICATION.

This thesis is dedicated to my wife, Felicia Elinam Dzamesi and daughter, Enam Ama Dzamesi.

KNUST



ABSTRACT

Dynamic programming is a useful mathematical technique for making a sequence of interrelated decisions. It provides a systematic procedure for determining the optimal combination of decisions. In contrast to linear programming, there does not exist a standard mathematical formulation of “the” dynamic programming problem. Rather, dynamic programming is a general type of approach to problem solving, and the particular equations used must be developed to fit each situation. Competitive order processing or product allocation that aims to deliver the required quantity of supply (goods) and services to the customers at the requested time demands precise planning and control mechanisms. The main objective of this research is to apply dynamic programming in solving the product allocation problem of a distribution company in Hohoe municipality. The aim is to minimize the cost of product allocation to customers or market centres and to have optional returns. And importantly to adopt more scientific approach in the running of the company in order to maximize returns. In solving the problem, dynamic programming algorithm and mathematical method formulated. Data collected from the company were used to solve the model using MATLAB and EXCEL of dynamic programming algorithm. It is observed from the analysis that the distribution company operates based on current market structure principles such as; if it closes down it loses its goodwill created, if it closes down it loses all the customers, with the hope that conditions may improve later, say in the long run for the firm to enjoy optimal return or at least breakeven and to maintain its skilled personnel. The company which has its optimal allocation policy based on the above market principles has optimal return of GH¢ 252,000.00. It is evident that, using dynamic programming approach, the optimum policy yields an optimum return of GH¢ 260,000.00. This thesis models product distribution problem as a dynamic programming problem. The model developed could be adopted for any problem that can be modelled as such.

TABLE OF CONTENTS

Contents	Page
Declaration	i
Acknowledgment	ii
Dedication	iii
Abstract	iv
Table of contents	v

CHAPTER 1 – INTRODUCTION

KNUST

1.0 Introduction.....	1
1.1 Background of Study.....	1
1.2 Problem Statement.....	7
1.3 Objectives.....	7
1.4 Methodology.....	8
1.5 Justification.....	8
1.6 Limitations of the Study.....	9
1.7 Organization of the Study.....	9
1.8 Summary.....	9

CHAPTER 2 – LITERATURE REVIEW

2.0 Introduction.....	10
2.1 Abstracts and review of Literature on Dynamic Programming.....	10

CHAPTER 3 – METHODOLOGY.

3.0	Introduction.....	49
3.1	Characteristics of Dynamic Programming Problems...	49
3.2	The Dynamic Programming Algorithm.....	51
3.3	The Formulation of Dynamic Programming Model.....	53

CHAPTER 4 – DATA COLLECTION AND ANALYSIS

4.0	Introduction.	54
4.1	Data Collection and Analysis.....	54
4.2	Results and Tables.....	56

CHAPTER 5 –CONCLUSIONS AND RECOMMENDATIONS

5.0	Introduction.....	64
5.1	Findings and Conclusions.....	64
5.2	Recommendations.....	65

REFERENCES...	66
---------------	-------	----

CHAPTER 1

1.0 INTRODUCTION

Dynamic programming is a very useful technique for making a sequence of interrelated decisions. It requires formulating an appropriate recursive relationship for each individual problem Eddy (2004). However, it provides a great computational savings over using exhaustive enumeration to find the best combination of decisions, especially for large problems. For example, if a problem has 10^3 stages with 10^3 states and 10^3 possible decisions at each stage, then exhaustive enumeration must consider up to 10^9 billion combinations, whereas dynamic programming need make no more than a thousand calculations (10^3 for each state at each stage). This study has considered only dynamic programming with a finite number of stages.

In this chapter of the thesis, we shall give an overview of dynamic programming model; a brief description of the problem statement of the thesis is also presented as well as the objectives, the methodology, the justification and the organization of the thesis.

1.1 BACKGROUND OF STUDY

Dynamic programming is a method for solving complex problems in mathematics and computer science by breaking them down into simpler sub problems. It is applicable to problems exhibiting the properties of overlapping, sub problems which are only slightly smaller and optimal substructure. When applicable, the method takes far less time than naive methods. The key idea behind dynamic programming is quite simple. In general, to solve a given problem, we need to solve different parts of the problem (sub problems), then combine the solutions of the sub problems to reach an overall solution. Often, many

of these sub problems are really the same. The dynamic programming approach seeks to solve each sub problem only once, thus reducing the number of computations.

This is especially useful when the number of repeating sub problems is exponentially large. The term dynamic programming was originally used in the 1940s by Richard Bellman to describe the process of solving problems where one needs to find the best decisions one after another. By 1953, the author refined this to the modern meaning, referring specifically to nesting smaller decision problems inside larger decisions, and the field was thereafter recognized by the Institute of Electronics and Electrical Engineering (IEEE) as a systems analysis and engineering topic. Bellman's contribution is remembered in the name of the Bellman equation, a central result of dynamic programming which restates an optimization problem in recursive form Bellman (1954).

The word dynamic was chosen by Bellman to capture the time-varying aspect of the problems, and also because it sounded impressive. The word programming referred to the use of the method to find an optimal program, in the sense of a military schedule for training or logistics Bellman (1957). Dynamic programming is both a mathematical optimization method and a computer programming method. In both contexts it refers to simplifying a complicated problem by breaking it down into simpler sub problems in a recursive manner.

While some decision problems cannot be taken apart this way, decisions that span several points in time do often break apart recursively; Bellman called this the "Principle of Optimality". Likewise, in computer science, a problem that can be broken down recursively is said to have optimal substructure. If sub problems can be nested recursively

inside larger problems, so that dynamic programming methods are applicable, then there is a relation between the value of the larger problem and the values of the sub problems. In the optimization literature this relationship is called the Bellman equation. 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, \dots, 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 $i=n-1, n-2 \dots 2, 1$ can be found by working backwards, using a recursive relationship called the Bellman equation. For $i = 2, \dots n$, 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 (www.wikipedia.org/wiki/dynamic_programing).

Dynamic programming is a useful mathematical technique for making a sequence of interrelated decisions. It provides a systematic procedure for determining the optimal combination of decisions. In contrast to linear programming, there does not exist a standard mathematical formulation of "the" dynamic programming problem. Rather, dynamic programming is a general type of approach to problem solving, and the particular equations used must be developed to fit each situation. Therefore, a certain

degree of ingenuity and insight into the general structure of dynamic programming problems is required to recognize when and how a problem can be solved by dynamic programming procedures. These abilities can best be developed by an exposure to a wide variety of dynamic programming applications and a study of the characteristics that are common to all these situations.

Many at times we may come across situations, where we may have to make decision in multistage, i.e. optimization of multistage decision problems. Dynamic programming is a technique for getting solutions for multistage decision problems. A problem, in which the decision has to be made at successive stages, is called a multistage decision problem. In this case, the problem solver will take decision at every stage, so that the total effectiveness defined over all the stages is optimal Bertsekas (2002).

Here the original problem is broken down or decomposed into small problems, which are known as sub problems or stages which is much convenient to handle and to find the optimal stage. For example, consider the problem of a sales manager, who wants to start from his head office and tour various branches of the company and reach the last branch. He has to plan his tour in such a way that he has to visit number of branches and cover less distance as far as possible. He has to divide the network of the route connecting all the branches into various stages and workout, which is the best route, which will help him to cover more branches and less distance. We can give plenty of business examples, which are multistage decision problems.

The computational technique used is known as Dynamic Programming or Recursive Optimization. We do not have a standard mathematical formulation of the Dynamic Programming Problem (D.P.P). For each problem, depending on the variables given, and

objective of the problem, one has to develop a particular equation to fit for situation Nocedal and Wright (2006).

Though we have quite good number of dynamic programming problems, sometimes to take advantage of dynamic programming, we introduce multistage nature in the problem and solve it by dynamic programming technique. Nowadays, application of Dynamic Programming is done in almost all day to day managerial problems, such as, inventory problems, waiting line problems, resource allocation problems etc.

Dynamic programming problem may be classified depending on the following conditions Kalavathy (2002);

(i) Dynamic programming problems may be classified depending on the nature of data available as Deterministic and Stochastic or Probabilistic models. In deterministic models, the outcome at any decision stage is unique, determined and known. In Probabilistic models, there is a set of possible outcomes with some probability distribution.

(ii) The possible decisions at any stage, from which we are to choose one, are called 'states'. These may be finite or infinite. States are the possible situations in which the system may be at any stage.

(iii) Total number of stages in the process may be finite or infinite and may be known or unknown.

Dynamic programming is a technique that can be used to solve many optimization problems. In most applications, dynamic programming obtains solutions by working

backward from the end of a problem toward the beginning, thus breaking up a large, unwieldy problem into a series of smaller, more tractable problems Kalavathy (2002).

One example of the usefulness of dynamic programming is the resource allocation problems. Resource allocation problems, in which limited resources must be allocated among several activities, are often solved by dynamic programming. Even though such problems can be solved by linear programming, for example, the Giapetto problem, to use linear programming to do resource allocation, one must make three assumptions Kalavathy (2002);

- (i) the amount of resource assigned to any activity may be any nonnegative value,
- (ii) the benefit obtained from each activity is proportional to the amount of the resource assigned to the activity, and
- (iii) the benefit obtained from more than one activity is the sum of the benefits obtained from the individual activities.

Even if assumptions 1 and 2 do not hold, dynamic programming can be used to solve resource-allocation problems efficiently when assumption 3 is valid and when the amount of the resource allocated to each activity is a member of a finite set.

In addition to the above application, dynamic programming have been used to solve a number of real-life problems, including network problems, equipment replacement problems, refinery capacity problems, travelling salesman problem. Thus dynamic has played an important role in supporting managerial decisions in the area of capital budgeting, warehouse location and scheduling.

1.2 PROBLEM STATEMENT

This thesis seeks to apply dynamic programming in solving the product allocation problem of a distribution company. Distribution companies normally have to decide the quantity of goods that should be sent to each market or consumption centers, considering the trade-off between being able to meet customer's satisfaction and minimizing the transportation cost from source to destination. The problem is a distribution of effort problem that has a linear objective function and a single constraint.

Fortunately, dynamic programming provides a solution with much less effort than exhaustive enumeration. (The computational savings are enormous for larger versions of this problem.) Dynamic programming starts with a small portion of the original problem and finds the optimal solution for this smaller problem. It then gradually enlarges the problem, finding the current optimal solution from the preceding one, until the original problem is solved in its entirety.

1.3 OBJECTIVES

The objectives of the study are:

- (i) to use dynamic programming for solving the product allocation problem of a distribution company in Hohoe municipality.
- (ii) to optimize the returns of a product allocation of a distribution company.

1.4 METHODOLOGY

For our methodology, we propose dynamic programming algorithm in solving our problem. First, the algorithm will be presented. A real life computational study will be performed.

1.5 JUSTIFICATION

Dynamic programming are widely used in financial decision making, and very interesting from the perspective of mathematical optimization and computer science because; it is able to simplify a complicated problem by breaking it down into simpler sub problems in a recursive manner. While some decision problems cannot be taken apart this way, decisions that span several points in time do often break apart recursively; Bellman called this the "Principle of Optimality". Likewise, in computer science, a problem that can be broken down recursively is said to have optimal substructure. If sub problems can be nested recursively inside larger problems, so that dynamic programming methods are applicable, then there is a relation between the value of the larger problem and the values of the sub problems.

In view of these, application of dynamic programming to solving real-life problems is an area of much interest in the contribution to academic knowledge, hence the reason for solving the dynamic programming problem.

1.6 LIMITATIONS OF THE STUDY

The main constraint of the study was inadequacy of funds. As a result, we decided not to consider a bigger company outside the municipality for the study. Hence, the study was restricted to a distribution company based in Hohoe township.

In addition to that, the company was not willing to disclose the actual returns from their transactions for security reasons and also for the fear that their tax returns may be increased.

1.7 ORGANIZATION OF THE THESIS

In Chapter 1, we presented a background study of dynamic programming. In Chapter 2, related works in the field dynamic programming will be reviewed. In Chapter 3, dynamic programming algorithm will be introduced and explained. Included in this Chapter will be a formal definition of the algorithm. Chapter 4 will provide a computational study of dynamic programming algorithm applied to real-life instances. Chapter 5 will conclude this thesis with additional comments on dynamic programming.

1.8 SUMMARY

In this chapter, the problem of optimal allocation of product of a distribution company is formulated. The objectives of the study were also stated. The history of Richard Bellman's principle of optimality was discussed briefly, and the overview of dynamic programming and its application to solving real life problems. In the next chapter, we shall review some relevant literature in the area of dynamic programming algorithm, model and applications.

CHAPTER 2

LITERATURE REVIEW

2.0 INTRODUCTION.

This Chapter seeks to review relevant literature related to the theory and applications of dynamic programming. It also critically examines the abstracts of various literature on deterministic, approximate and stochastic dynamic programming approach. And also, some literature on linear and non-linear programming were reviewed.

2.1 ABSTRACTS AND REVIEW OF LITERATURE ON DYNAMIC PROGRAMMING.

Steven (1975) presented a technique with the existence of an optimal stationary policy that can be obtained from the usual functional equation is again established in the presence of a bound (not necessarily polynomial) on the one-period reward of a semi-Markov decision process. This is done for both the discounted and the average cost case. In addition to allowing an uncountable state space, the law of motion of the system is rather general in that the author's model permits any state to be reached in a single transition. There is, however, a bound on a weighted moment of the next state reached. Finally, the author indicated the applicability of these results.

Dawen (1986) considered total reward Markov decision processes with countable state space using positive dynamic programming. For these models it is well known that in the positive case, i.e. the immediate reward function is nonnegative, without further conditions (i) the value iteration holds and (ii) there exist point wise good stationary

strategies. Here the author showed that (i) remains true if the non-negativity of the immediate reward function is replaced by the non-negativity of the value function and that (ii) remains true if there exists a strategy with nonnegative total rewards.

Held and Karp (1961) explored a dynamic programming approach to the solution of three sequencing problems: a scheduling problem involving arbitrary cost functions, the traveling-salesman problem, and an assembly line balancing problem. Each of the problems is shown to admit of numerical solution through the use of a simple recursion scheme; these recursion schemes also exhibit similarities and contrasts.

Yanhong and Scott (2002) described a systematic method for optimizing recursive functions using both indexed and recursive data structures. The method is based on two critical ideas: first, determining a minimal input increment operation so as to compute a function on repeatedly incremented input; second, determining appropriate additional values to maintain in appropriate data structures, based on what values are needed in computation on an incremented input and how these values can be established and accessed. Once these two are determined, the method extends the original program to return the additional values, derives an incremental version of the extended program, and forms an optimized program that repeatedly calls the incremental program. The method can derive all dynamic programming algorithms found in standard algorithm textbooks. There are many previous methods for deriving efficient algorithms, but none is as simple, general, and systematic as ours.

Mousavi and Karamouz (2003) developed a dynamic programming (DP) optimization model for long term planning of multiple-reservoir operations. To overcome the well-

known dimensionality problem associated with such a model, a heuristic approach is used to narrow the needed search algorithm within the state space of the DP model. This method can recognize many infeasible transitions from the initial to the final state of the DP stages. By diagnosing these infeasible transitions in advance and removing them from further computations, significant improvement in computational load is achieved so that the computer time for solving the model is reduced more than fifty (50) times for the reservoir system under study. This methodology is applied to a four-reservoir system located in Iran.

For G be an acyclic directed graph with weights and values assigned to its vertices. In the partially ordered knapsack problem we wish to find a maximum-valued subset of vertices whose total weight does not exceed a given knapsack capacity, and which contains every predecessor of a vertex if it contains the vertex itself. Johnson and Niemi (1983) considered the special case where G is an out-tree. Even though this special case is still NP-complete, we observe how dynamic programming techniques can be used to construct pseudo-polynomial time optimization algorithms and fully polynomial time approximation schemes for it. In particular, we show that a nonstandard approach we call “left-right” dynamic programming is better suited for this problem than the standard “bottom-up” approach, and we show how this “left-right” approach can also be adapted to the case of in-trees and to a related tree partitioning problem arising in integrated circuit design. We conclude by presenting complexity results which indicate that similar success cannot be expected with either problem when the restriction to trees is lifted.

The 0/1 knapsack (or knapsack without repetition) has a dynamic programming solution driven by a table in which each item is consecutively considered. Rolfe (2007) approached the problem by generating a table in which the optimal knapsack for each knapsack capacity is generated, modelled on the solution to the integer knapsack (knapsack with repetition) and the solution to change-making.

Alvarez et al., (1998) presented a model that uses dynamic programming in resolving economic issues may be extended to cover recursive methods, particularly homogenous problems. This may be made possible by defining the basic existence, uniqueness and convergence results generated from dynamic programming methods. The approach provides a concrete evidence for the Principle of Optimality, by showing that the dynamic program, itself, coincide accurately with the solutions of the original problem. The proposed strategy further provides a finite solution to the Bellman equation.

Charnes and Cooper (1956) presented a solution for the generalization of the warehouse model by means of dynamic programming techniques of one version of what is called the “warehouse problem”. The purpose of this note is to indicate how problems of this general nature may be approached by means of the functional equation technique of the theory of dynamic programming, and thereby reduced to a very simple and straightforward computational problem.

In several of the earliest literatures on dynamic programming (DP), reference was made to the possibility that the DP approach might be used to advise players on the optimal strategy for board games such as chess. Since these papers in the 1950s, there have been many attempts to develop such strategies, drawing on ideas from DP and other branches

of mathematics. Smith (2005) presented a survey of those where a dynamic programming approach has been useful, or where such a formulation of the problem will allow further insight into the optimal mode of play.

Sakoe (1978) studied an optimum dynamic programming (DP) based time-normalization algorithm for spoken word recognition. First, a general principle of time-normalization is given using time warping function. Then, two time-normalized distance definitions, symmetric and asymmetric forms, are derived from the principle. These two forms are compared with each other through theoretical discussions and experimental studies. The symmetric form algorithm superiority is established. A new technique, called slope constraint, is successfully introduced, in which the warping function slope is restricted so as to improve discrimination between words in different categories. Investigations were made, based on the assumption that speech patterns are time-sampled with a common and uniform sampling period, as in most general cases. One of the problems discussed in this paper involves the relative superiority of either a symmetric form of DP-matching or an asymmetric one. In the asymmetric form, time-normalization is achieved by transforming the time axis of a speech pattern onto that of the other. In the symmetric form, on the other hand, both time axes are transformed onto a temporarily defined common axis.

Theoretical and experimental comparisons show that the symmetric form gives better recognition than the asymmetric one. Another problem discussed concerns slope constraint technique. Since too much of the warping function flexibility sometimes results in poor discrimination between words in different the effective slope constraint characteristic is qualitatively analyzed, and the optimum slope constraint condition is

determined through experiments. The optimized algorithm is then extensively subjected to experimental comparison with various DP-algorithms, previously applied to spoken word recognition by different research groups. The experiment shows that the present algorithm gives no more than about two-thirds errors, even compared to the best conventional algorithm.

Zhang (2009) considered a nonlinear non-separable functional approximation to the value function of a dynamic programming formulation for the network revenue management (RM) problem with customer choice. We propose a simultaneous dynamic programming approach to solve the resulting problem, which is a nonlinear optimization problem with nonlinear constraints. We show that our approximation leads to a tighter upper bound on optimal expected revenue than some known bounds in the literature. Our approach can be viewed as a variant of the classical dynamic programming decomposition widely used in the research and practice of network RM. The computational cost of this new decomposition approach is only slightly higher than the classical version. A numerical study shows that heuristic control policies from the decomposition consistently outperform policies from the classical decomposition.

Liu (2004) presented an approach based on multi-scale representation and Dynamic Programming for matching deformed and possibly occluded shapes, which is robust with respect to noise and invariant to scale, translation, orientation and starting point selection. The process of contour segmentation can adjust automatically while the amounts of noise and deformation change. And the correspondence of similar parts of shapes helps to analyze the object structure and can be used as prior knowledge to learn shape model. We

have tested and evaluated our method on a database of one thousand, one hundred (1100) images of marine animals with a vast variety of shapes with very good results.

The query optimizer is one of the most important components of a database system. Most commercial query optimizers today are based on a dynamic-programming algorithm, as proposed in Selinger et al., (1979). While this algorithm produces good optimization results (i.e, good plans), its high complexity can be prohibitive if complex queries need to be processed, new query execution techniques need to be integrated, or in certain programming environments (e.g., distributed database systems). Donald and Konard (2000) presented and thoroughly evaluated a new class of query optimization algorithms that are based on a principle that we call iterative dynamic programming, or IDP for short. IDP has several important advantages: First, IDP-algorithms produce the best plans of all known algorithms in situations in which dynamic programming is not viable because of its high complexity. Second, some IDP variants are adaptive and produce as good plans as dynamic programming if dynamic programming is viable and as good-as possible plans if dynamic programming turns out to be not viable. Three, all IDP-algorithms can very easily be integrated into an existing optimizer which is based on dynamic programming.

Dynamic programming algorithms based on Lagrange multiplier method is often used for obtaining an optimal bit allocation strategy to minimize the total distortion given a constrained rate budget in both source and channel coding applications. Due to possible large quantizer set and improper initialization, the algorithm often suffers from heavy

computational complexity. There have been many solutions in recent years so the above question. YiSong et al., (2003) presented a simple but efficient algorithm to further speed up the convergence of the algorithm. This algorithm can be easily realized and get the final solution much faster. The experimental result shows that our new algorithm can figure out the optimal solution with a speed five-seven (5-7) times faster than the original algorithm.

Chisonge and Cole (2004) considered a network where nodes communicate by exchanging information packets whose fields include the address of the sending node and that of the destination node. In the absence of some verification mechanism, an attacking node can send packets to another node using a forged origin address. The author considered an optimization problem of identifying a minimum cardinality subset of verification nodes on a tree such that the number of attacks from any forged origin to any destination is limited to a prescribed level. For the case in which communication is permitted between every node in the tree, we develop an optimal polynomial-time dynamic programming algorithm for this problem. We compare the performance of the dynamic programming algorithm against a mixed-integer programming model on randomly generated tree networks at varied levels of security.

Matthew et al., (2007) presented an approximate dynamic programming approach for making ambulance redeployment decisions in an emergency medical service system. The primary decision is where we should redeploy idle ambulances so as to maximize the number of calls reached within a delay threshold. We begin by formulating this problem as a dynamic program. To deal with the high-dimensional and uncountable state space in the dynamic program, we construct approximations to the value function that are

parameterized by a small number of parameters. We tune the parameters using simulated cost trajectories of the system. Computational experiments demonstrate the performance of the approach on emergency medical service systems in two metropolitan areas. We report practically significant improvements in performance relative to benchmark static policies.

Dynamic programming (DP) is a popular technique which is used to solve combinatorial search and optimization problems. Guangming et al., (2009) studied a model that focused on one type of DP, which is called nonserial polyadic dynamic programming (NPDP). Owing to the nonuniform data dependencies of NPDP, it is difficult to exploit either parallelism or locality. Worse still, the emerging multi/many-core architectures with small on-chip memory make these issues more challenging.

In this paper, we address the challenges of exploiting the fine grain parallelism and locality of NPDP on multicore architectures. We describe a latency-tolerant model and a percolation technique for programming on multicore architectures. On an algorithmic level, both parallelism and locality do benefit from a specific data dependence transformation of NPDP. Next, we propose a parallel pipelining algorithm by decomposing computation operators and percolating data through a memory hierarchy to create just-in-time locality. In order to predict the execution time, we formulate an analytical performance model of the parallel algorithm. The parallel pipelining algorithm achieves not only high scalability on the 160-core IBM Cyclops64, but portable performance as well, across the 8-core Sun Niagara and quad-cores Intel Clovertown.

Ray directed volume-rendering algorithms are well suited for parallel implementation in a distributed cluster environment. For distributed ray casting, the scene must be partitioned between nodes for good load balancing, and a strict view-dependent priority order is required for image composition. Frank (2009) studied the load balanced network distribution (LBND) problem and map it to the NP-complete precedence constrained job-shop scheduling problem. We introduce a kd-tree solution and a dynamic programming solution. To process a massive data set, either a parallel or an out-of-core approach is required. Parallel preprocessing is performed by render nodes on data, which are allocated using a static data structure. Volumetric data sets often contain a large portion of voxels that will never be rendered, or empty space. Parallel preprocessing fails to take advantage of this. Our slab-projection slice, introduced in this paper, tracks empty space across consecutive slices of data to reduce the amount of data distributed and rendered. It is used to facilitate out-of-core bricking and kd-tree partitioning. Load balancing using each of our approaches is compared with traditional methods using several segmented regions of the Visible Korean data set.

Deependra (2010) proposed a framework that includes a penalty function incorporated stochastic dynamic programming (SDP) model in order to derive the operation policy of the reservoir of a hydropower plant, with an aim to reduce the amount of spill during operation of the reservoir. SDP models with various inflow process assumptions (independent and Markov-I) are developed and executed in order to derive the reservoir operation policies for the case study of a storage type hydropower plant located in Japan.

The policy thus determined consists of target storage levels (end-of-period storage levels) for each combination of the beginning-of-period storage levels and the inflow states of the current period. A penalty function is incorporated in the classical SDP model with objective function that maximizes annual energy generation through operation of the reservoir. Due to the inclusion of the penalty function, operation policy of the reservoir changes in a way that ensures reduced spill. Simulations are carried out to identify reservoir storage guide curves based on the derived operation policies. Reservoir storage guide curves for different values of the coefficient of penalty function, are plotted for a study horizon of sixty-four (64) years, and the corresponding average annual spill values are compared. It is observed that, with increasing values of the average annual spill decreases; however, the simulated average annual energy value is marginally reduced. The average annual energy generation can be checked vis-à-vis the average annual spill reduction, and the optimal value of, can be identified based on the cost functions associated with energy and spill.

Operation of a storage-based reservoir modifies the downstream flow usually to a value higher than that of natural flow in dry season. This could be important for irrigation, water supply, or power production as it is like an additional downstream benefit without any additional investment. Mahesh (2001) undertook a study that addresses the operation of two proposed reservoirs and the downstream flow augmentation at an irrigation project located at the outlet of the Gandaki River basin in Nepal. The optimal operating policies of the reservoirs were determined using a Stochastic dynamic programming (SDP) model

considering the maximization of power production. The modified flows downstream of the reservoirs were simulated by a simulation model using the optimal operating policy (for power maximization) and a synthetic long-term inflow series. Comparing the existing flow (flow in river without reservoir operation) and the modified flow (flow after reservoir operation) at the irrigation project, the additional amount of flow was calculated. The reliability analysis indicated that the supply of irrigation could be increased by twenty-five (25) to hundred (100) percent of the existing supply over the dry season (January to April) with a reliability of more than 80 percent.

A global mathematical model for simultaneously obtaining the optimal layout and design of urban drainage systems for foul sewage and storm water was presented by Freire (2000). The model can handle every kind of network, including parallel storm and foul sewers. It selects the optimal location for pumping systems and outfalls or wastewater treatment plants (defining the natural and artificial drainage basins), and it allows the presence of special structures and existing subsystems for optimal re-modelling or expansion. It is possible to identify two basic optimization levels: in the first level, the generation and transformation of general layouts (consisting of forests of trees) until a convergence criterion is reached, and in the second level, the design and evaluation of each forest. The global strategy adopted combines and develops a sequence of optimal design and plan layout sub-problems. Dynamic programming is used as a very powerful technique, alongside simulated annealing and genetic algorithms, in this discrete combinatorial optimization problem of huge dimension.

Shahid (2010) presented a new scheme for evaluating the performance of multithreaded computers and demonstrate its application to the Cray MTA-2 and XMT supercomputers. Our scheme is based on the concept of clock cycles per element, plotted against both problem size and the number of processors. This scheme showed that if an implementation has achieved its asymptotic efficiency and is more general than (but includes) the commonly used speedup metric. It permits the discovery of any imperfections in both the software as well as the hardware, and is expected to permit a unified comparison of much different parallel architecture. Measurements on a number of well-known parallel algorithms, ranging from matrix multiply to quicksort, are presented for the MTA-2 and XMT and highlight some interesting differences between these machines. The performance of sequence alignment using dynamic programming is evaluated on the MTA-2, XMT, IBM x3755 and SGI Altix 350 and provides a useful comparison of the capabilities of the Cray machines with more conventional shared memory architectures.

The notion of being sure that you have completely eradicated an invasive species is fanciful because of imperfect detection and persistent seed banks. Eradication is commonly declared either on an ad hoc basis, on notions of seed bank longevity, or on setting arbitrary thresholds of 1% or 5% confidence that the species is not present. Rather than declaring eradication at some arbitrary level of confidence, Tracey (2006) studied an economic approach in which we stop looking when the expected costs outweigh the expected benefits.

The author developed theory that determines the number of years of absent surveys required to minimize the net expected cost. Given detection of a species is imperfect, the

optimal stopping time is a trade-off between the cost of continued surveying and the cost of escape and damage if eradication is declared too soon.

A simple rule of thumb compares well to the exact optimal solution using stochastic dynamic programming. Application of the approach to the eradication programme of *Helenium amarum* reveals that the actual stopping time was a precautionary one given the ranges for each parameter.

Francisco (2010) examined the labour market effects of incomplete information about the workers' own job-finding process. Search outcomes convey valuable information, and learning from search generates endogenous heterogeneity in workers' beliefs about their job-finding probability. We characterize this process and analyze its interactions with job creation and wage determination. Our theory sheds new light on how unemployment can affect workers' labor market outcomes and wage determination, providing a rational explanation for discouragement as the consequence of negative search outcomes. In particular, longer unemployment durations are likely to be followed by lower reemployment wages because a worker's beliefs about his job-finding process deteriorate with unemployment duration. Moreover, our analysis provides a set of useful results on dynamic programming with optimal learning.

Gerardo (2008) presented a novel approach for capillary electrophoresis (CE) data analysis based on pattern recognition techniques in the wavelet domain. Low-resolution, denoised electropherograms are obtained by applying several preprocessing algorithms including denoising, baseline correction, and detection of the region of interest in the

wavelet domain. The resultant signals are mapped into character sequences using first derivative information and multilevel peak height quantization.

Next, a local alignment algorithm is applied on the coded sequences for peak pattern recognition. We also propose 2-D and 3-D representations of the found patterns for fast visual evaluation of the variability of chemical substances concentration in the analyzed samples. The proposed approach is tested on the analysis of intra-cerebral micro-dialysate data obtained by CE and LIF detection, achieving a correct detection rate of about 85% with a processing time of less than 0.3,s per 25,000-point electro-pherogram.

Using a local alignment algorithm on low-resolution denoised electropherograms might have a great impact on high-throughput CE since the proposed methodology will substitute automatic fast pattern recognition analysis for slow, human based time-consuming visual pattern recognition methods.

Masafumi (2010) studied the optimal operation of railway systems minimizing total energy consumption. Firstly, some measures of finding energy-saving train speed profiles are outlined. After the characteristics that should be considered in optimizing train operation are clarified, complete optimization based on optimal control theory is reviewed. Their basic formulations are summarized taking into account most of the difficult characteristics peculiar to railway systems.

Three methods of solving the formulation, dynamic programming (DP), gradient method, and sequential quadratic programming (SQP), are introduced. The last two methods can also control the state of charge (SOC) of the energy storage devices. By showing some numerical results of simulations, the significance of solving not only optimal speed profiles but also optimal SOC profiles of energy storage are emphasized, because the numerical results are beyond the conventional qualitative studies. Future scope for applying the methods to real-time optimal control is also mentioned.

Spjotvold (2009) considered the worst-case optimal control of discontinuous piecewise affine (PWA) systems, which are subjected to constraints and disturbances. The author seeks to pre-compute, via dynamic programming, an explicit control law for these systems when a PWA cost function is utilized. One difficulty with this problem class is that, even for initial states for which the value function of the optimal control problem is finite, there might not exist a control law that attains the infimum. Hence, we propose a method that is guaranteed to obtain a sub-optimal solution, and where the degree of sub-optimality can be specified a priori. This is achieved by approximating the underlying sub-problems with a parametric piecewise linear program.

When a hybrid electric vehicle (HEV) is certified for emissions and fuel economy, its power management system must be charge sustaining over the drive cycle, meaning that the battery state of charge (SOC) must be at least as high at the end of the test as it was at the beginning of the test. During the test cycle, the power management system is free to vary the battery SOC so as to minimize a weighted combination of fuel consumption and exhaust emissions. Edward (2007) presented a model which argued that shortest path stochastic dynamic programming (SP-SDP) offers a more natural formulation of the

optimal control problem associated with the design of the power management system because it allows deviations of battery SOC from a desired setpoint to be penalized only at key off.

This method is illustrated on a parallel hybrid electric truck model that had previously been analyzed using infinite-horizon stochastic dynamic programming with discounted future cost. Both formulations of the optimization problem yield a time-invariant causal state-feedback controller that can be directly implemented on the vehicle.

The advantages of the shortest path formulation include that a single tuning parameter is needed to trade off fuel economy and emissions versus battery SOC deviation, as compared with two parameters in the discounted, infinite-horizon case, and for the same level of complexity as a discounted future-cost controller, the shortest-path controller demonstrates better fuel and emission minimization while also achieving better SOC control when the vehicle is turned off. Linear programming is used to solve both stochastic dynamic programs.

The electric power industry is undergoing restructuring and deregulation. We need to incorporate the uncertainty of electric power demand or power generators into the unit commitment problem. The unit commitment problem is to determine the schedule of power generating units and the generating level of each unit.

The objective is to minimize the operational cost which is given by the sum of the fuel cost and the start-up cost. Takayuki (2004) presented a new algorithm for the stochastic unit commitment problem which is based on column generation approach. The algorithm continues adding schedules from the dual solution of the restricted linear master program

until the algorithm cannot generate new schedules. The schedule generation problem is solved by the calculation of dynamic programming on the scenario tree.

One partial solution to the problem of ever-increasing demands on our water resources is optimal allocation of available water. Bijan (2004) presented a non-linear programming (NLP) optimization model with an integrated soil water balance. This model is the advanced form of a previously developed one in which soil water balance was not included. The author proposed a dynamic programming approach for solving the problem. The model can perform over different crop growth stages while taking into account an irrigation time interval in each stage. Therefore, the results are directly applicable to real-world conditions. However, the time trend of actual evapo-transpiration (AET) for individual time intervals fluctuates more than that for growth-stage AETs. The proposed model was run for the Ardak area (45km NW of the city of Mashhad, Iran) under a single cropping cultivation (corn) as well as a multiple cropping pattern (wheat, barley, corn, and sugar beet). The water balance equation was manipulated with net applied irrigation water to overcome the difficulty encountered with incorrect deep percolation. The outputs of the model, under the imposed seasonal irrigation water shortages, were compared with the results obtained from a simple NLP model. The differences between these two models (simple and integrated) became more significant as irrigation water shortage increased. Oversimplified assumptions in the previous simple model were the main causes of these differences.

Real-time signal control operates as a function of the vehicular arrival and discharge process to satisfy a pre-specified operational performance. This process is often predicted based on loop detectors placed upstream of the signal. Fang (2010) developed a signal

control for diamond interchanges, a microscopic model to estimate traffic flows at the stop-line. The model considers the traffic dynamics of vehicular detection, arrivals, and departures, by taking into account varying speeds, length of queues, and signal control. As the signal control is optimized over a rolling horizon that is divided into intervals, the vehicular detection for and projection into the corresponding horizon intervals are also modeled. The signal control algorithm is based on dynamic programming and the optimization of signal policy is performed using a certain performance measure involving delays, queue lengths, and queue storage ratios. The arrival, discharge model is embedded in the optimization algorithm and both are programmed into AIMSUN, a microscopic stochastic simulation program. AIMSUN is then used to simulate the traffic flow and implement the optimal signal control by accessing internal data including detected traffic demand and vehicle speeds. Sensitivity analysis is conducted to study the effect of selecting different optimization criteria on the signal control performance. It is concluded that the queue length and queue storage ratio are the most appropriate performance measures in real-time signal control of interchanges.

Jushan (2003) considered practical issues for the empirical applications of the procedures. We first address the problem of estimation of the break dates and present an efficient algorithm to obtain global minimizers of the sum of squared residuals. This algorithm is based on the principle of dynamic programming and requires at most least-squares operations of order $O(T^2)$ for any number of breaks. Our method can be applied to both pure and partial structural change models. Second, we consider the problem of forming confidence intervals for the break dates under various hypotheses about the structure of

the data and the errors across segments. Third, we address the issue of testing for structural changes under very general conditions on the data and the errors. Fourth, we address the issue of estimating the number of breaks. Finally, a few empirical applications are presented to illustrate the usefulness of the procedures. All methods discussed are implemented in a GAUSS program.

An approximate dynamic programming (ADP) method has shown good performance in solving optimal control problems in many small-scale process control applications. The offline computational procedure of ADP constructs an approximation of the optimal "cost - to - go" function, which parameterizes the optimal control policy with respect to the state variable. With the approximate "cost - to - go" function computed, a multistage optimization problem that needs to be solved online at every sample time can be reduced to a single-stage optimization, thereby significantly lessening the real-time computational load. Thidarat (2009) addressed stochastic uncertainties within this framework. Nonetheless, the existing ADP method requires excessive offline computation when applied to a high-dimensional system. A case study of a reactor and a distillation column with recycle was used to illustrate this issue. Then, several ways were proposed to reduce the computational load so that the ADP method can be applied to high-dimensional integrated plants. The results showed that the approach is much more superior to NMPC in both deterministic and stochastic cases.

Optical microscopy allows a magnified view of the sample while decreasing the depth of focus. Although the acquired images from limited depth of field have both blurred and focused regions, they can provide depth information. The technique to estimate the depth

and 3D shape of an object from the images of the same sample obtained at different focus settings is called shape from focus (SFF). In SFF, the measure of focus, sharpness, is the crucial part for final 3D shape estimation. The conventional methods compute sharpness by applying focus measure operator on each 2D image frame of the image sequence. However, such methods do not reflect the accurate focus levels in an image because the focus levels for curved objects require information from neighboring pixels in the adjacent frames too. To address this issue, Seong (2009) proposed a new method based on focus adjustment which takes the values of the neighboring pixels from the adjacent image frames that have approximately the same initial depth as of the centre pixel and then it re-adjusts the center value accordingly. Experiments were conducted on synthetic and microscopic objects, and the results show that the proposed technique generates better shape and takes less computation time in comparison with previous SFF methods based on focused image surface (FIS) and dynamic programming.

An important technical component of natural resource management, particularly in an adaptive management context, is optimization. This is used to select the most appropriate management strategy, given a model of the system and all relevant available information. For dynamic resource systems, dynamic programming has been the de facto standard for deriving optimal state-specific management strategies. Though effective for small-dimension problems, dynamic programming is incapable of providing solutions to larger problems, even with modern micro-computing technology. Reinforcement learning is an alternative, related procedure for deriving optimal management strategies, based on stochastic approximation. It is an iterative process that improves estimates of the value of

state-specific actions based in interactions with a system, or model thereof. Applications of reinforcement learning in the field of artificial intelligence have illustrated its ability to yield near-optimal strategies for very complex model systems, highlighting the potential utility of this method for ecological and natural resource management problems, which tend to be of high dimension. The author described the concept of reinforcement learning and its approach of estimating optimal strategies by temporal difference learning. He then illustrate the application of this method using a simple, well-known case study of Anderson [1975], and compare the reinforcement learning results with those of dynamic programming. Though a globally-optimal strategy is not discovered, it performs very well relative to the dynamic programming strategy, based on simulated cumulative objective return. Christopher (2005) suggested that reinforcement learning be applied to relatively complex problems where an approximate solution to a realistic model is preferable to an exact answer to an oversimplified model.

Approximate dynamic programming (ADP) is a broad umbrella for a modeling and algorithmic strategy for solving problems that are sometimes large and complex, and are usually (but not always) stochastic. It is most often presented as a method for overcoming the classic curse of dimensionality that is well-known to plague the use of Bellman's equation. For many problems, there are actually up to three curses of dimensionality. But the richer message of approximate dynamic programming is learning what to learn, and how to learn it, to make better decisions over time. Warren (2009) presented a brief review of approximate dynamic programming, without intending to be a complete

tutorial. Instead, our goal is to provide a broader perspective of ADP and how it should be approached from the perspective of different problem classes.

Stochastic dynamic programming models are attractive for multi-reservoir control problems because they allow non-linear features to be incorporated and changes in hydrological conditions to be modelled as Markov processes. However, with the exception of the simplest cases, these models are computationally intractable because of the high dimension of the state and action spaces involved. Archibald (2006) proposed a new method of determining an operating policy for a multi-reservoir control problem that uses stochastic dynamic programming, but is practical for systems with many reservoirs. Decomposition is first used to reduce the problem to a number of independent subproblems. Each subproblem is formulated as a low-dimensional stochastic dynamic program and solved to determine the operating policy for one of the reservoirs in the system.

Cheng-Liang Chen (2003) proposed a novel algorithm integrating iterative dynamic programming and fuzzy aggregation to solve multi-objective optimal control problems. First, the optimal control policies involving these objectives are sequentially determined. A payoff table is then established by applying each optimal policy in series to evaluate these multiple objectives. Considering the imprecise nature of decision-maker's judgment, these multiple objectives are viewed as fuzzy variables. Simple monotonic increasing or decreasing membership functions are then defined for degrees of satisfaction for these linguistic objective functions. The optimal control policy is finally searched by maximizing the aggregated fuzzy decision values. The proposed method is

rather easy to implement. Two chemical processes, Nylon 6 batch polymerization and Penicillin G fed-batch fermentation, are used to demonstrate that the method has a significant potential to solve real industrial problems.

The Cramér-Lundberg insurance model is studied where the risk process can be controlled by reinsurance and by investment in a financial market. The performance criterion is the ruin probability. Manfred (2003) studied this problem by can imbedding in the framework of discrete-time stochastic dynamic programming. Basic tools are the Howard improvement and the verification theorem. Explicit conditions are obtained for the optimality of employing no reinsurance and of not investing in the market.

The accuracy of an alignment between two protein sequences can be improved by including other detectably related sequences in the comparison. Marc (2004) optimized and benchmarked such an approach that relies on aligning two multiple sequence alignments, each one including one of the two protein sequences. Thirteen(13) different protocols for creating and comparing profiles corresponding to the multiple sequence alignments are implemented in the SALIGN command of MODELLER. A test set of 200 paire wise, structure-based alignments with sequence identities below 40% is used to benchmark the thirteen (13) protocols as well as a number of previously described sequence alignment methods, including heuristic pairwise sequence alignment by BLAST, pairwise sequence alignment by global dynamic programming with an affine gap penalty function by the ALIGN command of MODELLER, sequence-profile alignment by PSI-BLAST, Hidden Markov Model methods implemented in SAM and LOBSTER, pairwise sequence alignment relying on predicted local structure by SEA,

and multiple sequence alignment by CLUSTALW and COMPASS. The alignment accuracies of the best new protocols were significantly better than those of the other tested methods. For example, the fraction of the correctly aligned residues relative to the structure-based alignment by the best protocol is 56%, which can be compared with the accuracies of 26%, 42%, 43%, 48%, 50%, 49%, 43%, and 43% for the other methods, respectively. The new method is currently applied to large-scale comparative protein structure modeling of all known sequences.

Jacoboni (2001) presented a method based on neural networks and tested on a non-redundant set of barrel membrane proteins known at atomic resolution with a jackknife procedure. The method predicts the topography of trans-membrane, strands with residue accuracy as high as 78% when evolutionary information is used as input to the network. Of the trans-membrane, strands included in the training set, 93% are correctly assigned. The predictor includes an algorithm of model optimization, based on dynamic programming that correctly models eight out of the eleven (11) proteins present in the training/testing set. In addition, protein topology is assigned on the basis of the location of the longest loops in the models. The author proposed this as a general method to fill the gap of the prediction of, barrel membrane proteins.

Tsai (2005) presented an automatic and more robust implementation of multivariate adaptive regression splines (MARS) within the orthogonal array (OA)/MARS continuous-state stochastic dynamic programming (SDP) method. MARS is used to estimate the future value functions in each SDP level. The default stopping rule of MARS

employs the maximum number of basis functions M_{\max} , specified by the user. To reduce the computational effort and improve the MARS fit for the wastewater treatment SDP model, two automatic stopping rules, which automatically determine an appropriate value for M_{\max} , and a robust version of MARS that prefers lower-order terms over higher-order terms are developed. Computational results demonstrate the success of these approaches.

In solving the boundary value problem resulting from the use of Pontryagin's maximum principle, a transformation matrix is used to relate the sensitivity of the final state to the initial state. This avoids the need to solve the $(n \times n)$ differential equation to give the transition matrix, and yields very rapid convergence to the optimum. To ensure convergence, Rein (2010) proposed an iterative dynamic programming (IDP) for a number of passes to yield good starting conditions for this boundary condition iteration procedure. Clipping technique is used to handle constraints on control. Five optimal control problems were used to illustrate and to test the procedure.

At times, the objective is to seek a bang-bang control policy for nonlinear time-optimal control problems. The usefulness of iterative dynamic programming (IDP) has been shown in the literature for solving such problems. However, the convergence to the optimal solution has been obtained from about 50% of the guessed values near the optimum. Yash (2000) presented an improved IDP search method for seeking such solutions and a comparison is made with the IDP. The results show that the convergence can be obtained from a significantly higher number of guessed values chosen over a much wider region around the optimum.

The discretized quadratic sub-optimal tracker for nonlinear continuous two-dimensional (2-D) systems is newly proposed by Chia-Wei (2004). The proposed method provides a novel methodology for indirect digital redesign for nonlinear continuous 2-D systems with a continuous performance index. This includes the following features: (i) the 2-D optimal-linearization approach of the nonlinear 2-D Roesser's model (RM), (ii) the dynamic programming-based discretized quadratic optimal tracker for linear continuous 2-D systems, (iii) the steady-state discretized quadratic sub-optimal tracker for linear continuous 2-D systems, and (iv) the discretized quadratic sub-optimal tracker for nonlinear continuous 2-D systems. Illustrative examples were presented to demonstrate the effectiveness of the proposed procedure.

A symbolic dynamic programming approach for modelling first-order Markov decision processes within the fluent calculus was studied by Grobmann et al., (2002). Based on an idea initially presented, the major components of Markov decision processes such as the optimal value function and a policy are logically represented. The technique produces a set of first-order formulae with equality that minimally partitions the state space. Consequently, the symbolic dynamic programming algorithm presented here does not require to enumerate the state and action spaces, thereby solving a drawback of classical dynamic programming methods. In addition, we illustrate how conditional actions and specificity can be modelled by the approach.

The curse of dimensionality gives rise to prohibitive computational requirements that render infeasible the exact solution of large-scale stochastic control problems. De Farias

and Dawen (2001) studied an efficient method based on dynamic programming for approximating solutions to such problems. The approach “fits” a linear combination of pre-selected basis functions to the dynamic programming cost-to-go function. The authors developed error bounds that offer performance guarantees and also guide the selection of both basis functions and “state-relevance weights” that influence quality of the approximation. Experimental results in the domain of queuing network control provide empirical support for the methodology.

Bertossi and Mei (2000) presented several dynamic programming algorithms which can be efficiently implemented using parallel networks with reconfigurable buses. The bit model of general reconfigurable meshes with directed links, common write, and unit-time delay for broadcasting is assumed. Given two sequences of length m and n , respectively, their longest common subsequence can be found in constant time by an $O(mh) \cdot O(nh)$ directed reconfigurable mesh, where $h = \min\{m, n\} + 1$. Moreover, given an n -node directed graph $G = (V, E)$ with (possibly negative) integer weights on its arcs, the shortest distances from a source node $v \in V$ to all other nodes can be found in constant time by an $O(n^2w) \times O(n^2w)$ directed reconfigurable mesh, where w is the maximum weight.

Godfrey and Powell (2000) studied an adaptive dynamic programming algorithm for stochastic dynamic resource allocation problems, which arise in the context of logistics and distribution, fleet management, and other allocation problems. The method depends on estimating separable nonlinear approximations of value functions, using a dynamic programming framework. That paper considered only the case in which the time to

complete an action was always a single time period. Experiments with this technique quickly showed that when the basic algorithm was applied to problems with multi-period travel times, the results were very poor. In this paper, we illustrate why this behavior arose, and propose a modified algorithm that addresses the issue. Experimental work demonstrates that the modified algorithm works on problems with multiperiod travel times, with results that are almost as good as the original algorithm applied to single period travel times.

KNUST

The most common application of linear programming in agricultural situations has been to the problem of resource allocation between competing farm activities. Given relevant input-output information for a specific farm, together with real or assumed price and cost patterns, the technique of linear programming enables calculation of the combination of enterprises which maximizes net profit, within the limitations imposed by the availability of farm resources. It is necessary in some linear programming analyses to make explicit allowance for the peculiar influence of time on the structure of the system under study. Of the many ways in which this may be achieved, Throsby (1962) studied four proposals, which have been, or are likely to be, of relevance in an agricultural context: (i) Parametric programming, which allows consideration of resource or price variation between time periods; (ii) extension of the time-span of an activity to cover a series of sequential processes, for example the treatment of rotational sequences as single activities; (iii) the referencing of some resources and/or activities to specific time periods; a common example is the fragmentation of labour supply into months; and (iv) the so-called "multi-stage" or "dynamic" linear programming where a single matrix is used to describe, in an orderly fashion, a system's structure over a time-span of several periods. It

is the latter with which we are primarily concerned here. In its simplest form a dynamic linear programming problem may be set up as a large matrix composed of a series of smaller matrices lying down the diagonal. In its more advanced form allowance can be made for interactions between resources and activities in different periods. In general, dynamic linear programming problems are characterized by large "sparse" matrices (i.e., matrices in which many coefficients are zero) and usually a "block diagonal" or "block triangular" pattern is evident. The size of such matrices is frequently forbidding; however, computational algorithms are available which allow overall solutions to be obtained by solving a series of smaller problems. With the aid of a little ingenuity a great variety of time-dependent restrictions, resources, activities and opportunities can be accounted for in a dynamic linear programming analysis. From an agricultural economist's viewpoint it would not seem extravagant to claim that dynamic linear programming can be used to provide a more adequate analytical description of whole-farm situations over time than most other tools at present available in his kit.

Milios and Petrakis (1999) presented a shape matching algorithm for deformed shapes based on dynamic programming. Our algorithm is capable of grouping together segments at finer scales in order to come up with appropriate correspondences with segments at coarser scales. The authors illustrated the effectiveness of our algorithm in retrieval of shapes by content on two different two-dimensional (2-D) datasets, one of static hand gesture shapes and another of marine life shapes. The authors also demonstrated the superiority of their approach over traditional approaches to shape matching and retrieval, such as Fourier descriptors and geometric and sequential moments. Our evaluation is

based on human relevance judgments following a well-established methodology from the information retrieval field.

Tohru (2007) studied capacity expansion problems for telecommunication network facilities, based on fuzzy dynamic programming. Although cost functions, discount rates, demand functions, and so on should be given, they usually cannot be defined clearly because of technical developments or business fluctuations. This paper represents undefined factors by fuzzy numbers with triangular membership functions (TMF). Multiplication or division of TMF does not give rigid TMF, but we approximate them to TMF for ease of calculation. Three methods based on this approximation are compared, using numerical examples. Approximation accuracies are confirmed by strict calculation using removal of defining orders of fuzzy numbers.

An investigation of the single-vehicle, many-to-many, immediate-request dial-a-ride problem was developed in two parts (I and II) by Harilaos (1980). Part I focuses on the “static” case of the problem. In this case, intermediate requests that may appear during the execution of the route are not considered. A generalized objective function is examined, the minimization of a weighted combination of the time to service all customers and of the total degree of “dissatisfaction” experienced by them while waiting for service. This dissatisfaction is assumed to be a linear function of the waiting and riding times of each customer. Vehicle capacity constraints and special priority rules are part of the problem. A Dynamic Programming approach was proposed. The algorithm exhibits a computational effort which, although an exponential function of the size of the problem, is asymptotically lower than the corresponding effort of the classical Dynamic

Programming algorithm applied to a Traveling Salesman Problem of the same size. Part II extends this approach to solving the equivalent “dynamic” case. In this case, new customer requests are automatically eligible for consideration at the time they occur. The procedure is an open-ended sequence of updates, each following every new customer request. The algorithm optimizes only over known inputs and does not anticipate future customer requests. Indefinite deferment of a customer’s request is prevented by the priority rules introduced in Part I. Examples in both “static” and “dynamic” cases are presented.

Dan (2009) considered a nonlinear non-separable functional approximation to the value function of a dynamic programming formulation for the network revenue management (RM) problem with customer choice. The authors proposed a simultaneous dynamic programming approach to solve the resulting problem, which is a nonlinear optimization problem with nonlinear constraints. We show that our approximation leads to a tighter upper bound on optimal expected revenue than some known bounds in the literature. Our approach can be viewed as a variant of the classical dynamic programming decomposition widely used in the research and practice of network RM. The computational cost of this new decomposition approach is only slightly higher than the classical version. A numerical study shows that heuristic control policies from the decomposition consistently outperform policies from the classical decomposition.

To reduce delay in ship operations in automated container terminals, it is important to make different types of container handling equipment to operate harmoniously during this operation. Delivery operations by automated guided vehicles (AGVs) play an important role for synchronizing operations of container cranes with yard cranes. Kap

and Jong (2000) studied how to dispatch AGVs by utilizing information about locations and times of future delivery tasks. A mixed-integer programming model was provided for assigning optimal delivery tasks to AGVs. A heuristic algorithm is suggested for overcoming the excessive computational time needed for solving the mathematical model. Objective values and computational times of the heuristic algorithm are compared with those of the optimizing method. To test performances of the heuristic algorithm, a simulation study is performed by considering the uncertainties of various operation times and the number of future delivery tasks for looking ahead. Also, the performance of the heuristic algorithm is compared with those of other dispatching rules.

Car pooling is a transportation service organized by a large company which encourages its employees to pick up colleagues while driving to/from work to minimize the number of private cars travelling to/from the company site. The car pooling problem consists of defining the subsets of employees that will share each car and the paths the drivers should follow, so that sharing is maximized and the sum of the path costs is minimized. The special case of the car pooling problem where all cars are identical can be modeled as a Dial-a-Ride Problem. Roberto et al., (2000) presented a dynamic programming model for the car pooling problem, based on integer programming formulations of the problem. The method was based on a bounding procedure that combines three lower bounds derived from different relaxations of the problem. A valid upper bound is obtained by a heuristic method, which transforms the solution of a Lagrangean lower bound into a feasible solution. The computational results show the effectiveness of the proposed methods.

The Traveling Salesman Problem with Time Windows (TSPTW) is the problem of finding a minimum-cost path visiting a set of cities exactly once, where each city must be

visited within a specific time window. Filippo et al., (2001) presented a dynamic programming approach for solving the TSPTW that merges Constraint Programming propagation algorithms for the feasibility viewpoint (find a path), and Operations Research techniques for coping with the optimization perspective (find the best path). The authors showed with extensive computational results that the synergy between Operations Research optimization techniques embedded in global constraints, and Constraint Programming constraint solving techniques, makes the resulting framework effective in the TSPTW context also if these results are compared with state-of-the-art algorithms from the literature.

Dynamic programming solutions to a number of different recurrence equations for sequence comparison and for RNA secondary structure prediction were considered by Eppstein et al., (1992). These recurrences are defined over a number of points that is quadratic in the input size; however only a sparse set matters for the result. Efficient algorithms for these problems are given, when the weight functions used in the recurrences are taken to be linear. The time complexity of the algorithms depends almost linearly on the number of points that need to be considered; when the problems are sparse this results in a substantial speed-up over known algorithms.

Andrew et al., (1997) developed a dynamic programming based system for managing inventory at Jeppesen Sanderson, Inc., a major provider of aviation-information products. The system determines order quantities for charts used in flight manuals. These charts contain essential safety information that changes frequently, making standard methods for inventory management ineffective. The formulated the problem as a dynamic

programming model and developed a simple heuristic-solution procedure for determining order quantities. Based on this procedure, we also developed a decision support system (DSS) and implemented it for 600 of the most expensive Jeppesen charts. The system has been in use since August 1998, generating actual annual cost reductions of over \$800,000.

Current methods for identification of potential triplex-forming sequences in genomes and similar sequence sets rely primarily on detecting homopurine and homopyrimidine tracts. Procedures capable of detecting sequences supporting imperfect, but structurally feasible intra-molecular triplex structures are needed for better sequence analysis. Matej et al., (2010) presented a dynamic programming algorithm for detection of approximate palindromes, so as to account for the special nature of triplex DNA structures. From available literature, we conclude that approximate triplexes tolerate two classes of errors. One, analogical to mismatches in duplex DNA, involves nucleotides in triplets that do not readily form Hoogsteen bonds. The other class involves geometrically incompatible neighboring triplets hindering proper alignment of strands for optimal hydrogen bonding and stacking. We tested the statistical properties of the algorithm, as well as its correctness when confronted with known triplex sequences. The proposed algorithm satisfactorily detects sequences with intra-molecular triplex-forming potential. Its complexity is directly comparable to palindrome searching.

The substitution rate in a gene can provide valuable information for understanding its functionality and evolution. A widely used method to estimate substitution rates is the maximum-likelihood method implemented in the CODEML program in the PAML package. A limited number of branch models, chosen based on a priori information or an

interest in a particular lineage(s), are tested, whereas a large number of potential models are neglected. A complementary approach is also needed to test all or a large number of possible models to search for the globally optimal model(s) of maximum likelihood. However, the computational time for this search even in a small number of sequences becomes impractically long. Thus, it is desirable to explore the most probable spaces to search for the optimal models. Using dynamic programming techniques, Chengjun et al., (2010) developed a simple computational method for searching the most probable optimal branch-specific models in a practically feasible computational time. We propose three search methods to find the optimal models, which explored $O(n)$ (method 1) to $O(n^2)$ (method 2 and method 3) models when the given phylogeny has n branches. In addition, we derived a formula to calculate the number of all possible models, revealing the complexity of finding the optimal branch-specific model. We show that in a reanalysis of over fifty (50) previously published studies, the vast majority obtained better models with significantly higher likelihoods than the conventional hypothesis model methods.

Allocating water between different users and uses, including the environment, is one of the most challenging task facing water resources managers and has always been at the heart of Integrated Water Resources Management (IWRM). As water scarcity is expected to increase over time, allocations decisions among the different uses will have to be found taking into account the complex interactions between water and the economy. Hydro-economic optimization models can capture those interactions while prescribing efficient allocation policies. Many hydro-economic models found in the literature are formulated as large-scale non linear optimization problems (NLP), seeking to maximize

net benefits from the system operation while meeting operational and/or institutional constraints, and describing the main hydrological processes. However, those models rarely incorporate the uncertainty inherent to the availability of water, essentially because of the computational difficulties associated stochastic formulations. Goor et al., (2008) presented a dynamic programming model that can identify economically efficient allocation policies in large-scale multipurpose multireservoir systems. The model is based on stochastic dual dynamic programming (SDDP), an extension of traditional SDP that is not affected by the curse of dimensionality. SDDP identify efficient allocation policies while considering the hydrologic uncertainty. The objective function includes the net benefits from the hydropower and irrigation sectors, as well as penalties for not meeting operational and/or institutional constraints. To be able to implement the efficient decomposition scheme that remove the computational burden, the one-stage SDDP problem has to be a linear program. Recent developments improve the representation of the non-linear and mildly non-convex hydropower function through a convex hull approximation of the true hydropower function. This model is illustrated on a cascade of fourteen (14) reservoirs on the Nile river basin.

Hybrid electric vehicles (HEVs) combined with more than one power source offer additional flexibility to improve the fuel economy and to reduce pollutant emissions. The dynamic-programming-based supervisory controller (DPSC) was studied by G-Qaoi et al., (2008) which investigates the fuel economy improvement and emissions reduction potential and demonstrates the trade-off between fuel economy and the emission of nitrogen oxides (NO_x) for a state-of-charge-sustaining parallel HEV. A weighted cost function consisting of fuel economy and emissions is proposed in this paper. Any

possible engine-motor power pairs meeting with the power requirement is considered to minimize the weighted cost function over the given driving cycles through this dynamic program algorithm. The fuel-economy-only case, the NO_x -only case, and the fuel- NO_x case have been achieved by adjusting specific weighting factors, which demonstrates the flexibility and advantages of the DPSC. Compared with operating the engine in the NO_x -only case, there is 17.4 per cent potential improvement in the fuel-economy-only case. The fuel- NO_x case yields a 15.2 per cent reduction in NO_x emission only at the cost of 5.5 per cent increase in fuel consumption compared with the fuel-economy-only case.

Khaneja et al., (1988) presented Dynamic programming algorithms for automated generation of length minimizing geodesics and curves of extremal curvature on the neocortex of the macaque and the Visible Human. Probabilistic models of curve variation are constructed in terms of the variability in speed, curvature, and torsion in the Frenet representation.

In cricket, when a batsman is dismissed towards the end of a day's play, he is often replaced by a lower-order batsman (a 'night watchman'), in the hope that the remaining recognised batsmen can start their innings on the following day. Clarke and Norman (2003) studied a dynamic programming analysis which suggests that the common practice of using a lower-order batsman is often sub-optimal. Towards the end of a day's play, when the conventional wisdom seems to be to use a night watchman, it may be best to send in the next recognised batsman in the batting order. Sending in a night watchman may be good judgement when there are several recognised batsman and several lower order batsmen still to play (say four of each). However, with smaller numbers (two of

each, for example), then, with very few overs left to play, it may be better to send in a recognised batsman.

Rust (1987) presented a model of retirement behavior based on the solution to a dynamic programming problem. The workers objective is to maximize expected discounted utility over his remaining lifetime. At each time period the worker chooses how much to consume and whether to work full-time, part-time, or exit the labor force. The model accounts for the sequential nature of the retirement decision problem, and the role of expectations of uncertain future variables such as the worker's future lifespan, health status, marital and family status, employment status, as well as earnings from employment, assets, and social security retirement, disability and Medicare payments. This method applies a "nested fixed point" algorithm that converts the dynamic programming problem into the problem of repeatedly re-computing the fixed point to a contraction mapping operator as a subroutine of a standard nonlinear maximum likelihood program. The goal of the paper is to demonstrate that a fairly complex and realistic formulation of the retirement problem can be estimated using this algorithm and a current generation supercomputer, the Cray-2.

CHAPTER 3

METHODOLOGY

3.0 INTRODUCTION

This chapter provides an in depth explanation of the dynamic programming.

In order to understand the value of dynamic programming, it is necessary to have a good understanding of some key terms as used in dynamic programming problems

KNUST

3.1 CHARACTERISTICS OF DYNAMIC PROGRAMMING PROBLEMS

One way to recognize a situation that can be formulated as a dynamic programming problem is to notice that its basic features.

These basic features that characterize dynamic programming problems are presented and discussed here.

(i) The problem can be divided into stages, with a policy decision required at each stage.

Dynamic programming problems require making a sequence of interrelated decisions, where each decision corresponds to one stage of the problem.

(ii) Each stage has a number of states associated with the beginning of that stage.

In general, the states are the various possible conditions in which the system might be at that stage of the problem. The number of states may be either finite or infinite.

(iii) The effect of the policy decision at each stage is to transform the current state to a state associated with the beginning of the next stage (possibly according to a probability distribution).

This procedure suggests that dynamic programming problems can be interpreted in terms of the networks. Each node would correspond to a state. The network would consist of

columns of nodes, with each column corresponding to a stage, so that the flow from a node can go only to a node in the next column to the right. The links from a node to nodes in the next column correspond to the possible policy decisions on which state to go to next. The value assigned to each link usually can be interpreted as the immediate contribution to the objective function from making that policy decision. In most cases, the objective corresponds to finding either the shortest or the longest path through the network.

(iv) The solution procedure is designed to find an optimal policy for the overall problem, i.e., a prescription of the optimal policy decision at each stage for each of the possible states.

For any problem, dynamic programming provides this kind of policy prescription of what to do under every possible circumstance (which is why the actual decision made upon reaching a particular state at a given stage is referred to as a policy decision). Providing this additional information beyond simply specifying an optimal solution (optimal sequence of decisions) can be helpful in a variety of ways, including sensitivity analysis.

(v) Given the current state, an optimal policy for the remaining stages is independent of the policy decisions adopted in previous stages. Therefore, the optimal immediate decision depends on only the current state and not on how you got there. This is the principle of optimality for dynamic programming.

For dynamic programming problems in general, knowledge of the current state of the system conveys all the information about its previous behavior necessary for determining the optimal policy henceforth. Any problem lacking this property cannot be formulated as a dynamic programming problem.

(vi) The solution procedure begins by finding the optimal policy for the last stage.

The optimal policy for the last stage prescribes the optimal policy decision for each of the possible states at that stage. The solution of this one-stage problem is usually trivial, as it was for the stagecoach problem.

(vii) A recursive relationship that identifies the optimal policy for stage n , given the optimal policy for stage $n + 1$, is available.

Therefore, finding the optimal policy decision when you start in state s at stage n requires finding the minimizing value of x_n .

This property is emphasized in the next (and final) characteristic of dynamic programming.

(viii) When we use this recursive relationship, the solution procedure starts at the end and moves backward stage by stage - each time finding the optimal policy for that stage - until it finds the optimal policy starting at the initial stage. This optimal policy immediately yields an optimal solution for the entire problem.

3.2 THE DYNAMIC PROGRAMMING ALGORITHM

- Identify the decision variables and specify objective function to be optimized under certain limitations, if any.
- Decompose or divide the given problem into a number of smaller sub-problems or stages. Identify the state variables at each stage and write down the transformation function as a function of the state variable and decision variables at the next stage.
- Write down the general recursive relationship for computing the optimal policy. Decide whether forward or backward method is to follow to solve the problem.

- Construct appropriate stage to show the required values of the return function at each stage.
- Determine the overall optimal policy or decisions and its value at each stage. There may be more than one such optimal policy.

The basic features, which characterize the dynamic programming problem, are as follows:

- (i) Problem can be sub-divided into stages with a policy decision required at each stage. A stage is a device to sequence the decisions. That is, it decomposes a problem into sub-problems such that an optimal solution to the **problem** can be obtained from the optimal solution to the sub-problem.
- (ii) Every stage consists of a number of states associated with it. The states are the different possible conditions in which the **system** may find itself at that stage of the problem.
- (iii) Decision at each stage converts the **current stage** into state associated with the next stage.
- (iv) The state of the system at a stage is described by a set of variables, called state variables.
- (v) When the current state is known, an optimal policy for the remaining stages is independent of the policy of the previous ones.
- (vi) To identify the optimum policy for each state of the system, a recursive equation is formulated with ' n ' stages remaining, given the optimal policy for each stage with $(n - 1)$ stages left.

(vii) Using recursive equation approach each time the solution procedure moves backward, stage by stage for obtaining the optimum policy of each stage for that particular stage, still it attains the optimum policy beginning at the initial stage.

3.3 THE FORMULATION OF DYNAMIC PROGRAMMING MODEL

The formulation for solving multistage problem is as follows;

Let the decision variables $x_n (n = 1, 2, 3, \dots, N)$ be the immediate destination on stage n (the n th stage problem to be solved). Thus, the problem selected is $x_1 \rightarrow x_2 \rightarrow x_3 \dots \rightarrow x_n$, where $x_n \rightarrow J$, and J is the ultimate destination.

Let $f_n(s, x_n)$ be the total cost of the best overall policy for the remaining stages, given that we are in state s , ready to start stage n , and selects x_n as the immediate destination.

Given s and n ,

let x_n^* denote any value of x_n (not necessarily unique) that minimizes $f_n(s, x_n)$, and let $f_n^*(s)$ be the corresponding minimum value of $f_n(s, x_n)$. Thus,

$$f_n^*(s) = \min f_n(s, x_n) = f_n(s, x_n^*), \text{ where}$$

$$f_n(s, x_n) = \text{immediate cost (stage } n) + \text{minimum future cost (stage } n+1 \text{ onward)} = c_{s,x_n} + f_{n+1}^*(x_n).$$

The value of c_{s,x_n} is given by setting $i = s$ (the current state) and $j = x_n$ (the immediate destination). Because the ultimate destination (state J) is reached at the end of stage n ,

$$f_{n+1}^*(J) = 0.$$

The objective is to find $f^*(i(A))$ and the corresponding route. Dynamic programming finds it by successively finding $f_n^*(s), f_{n-1}^*(s), f_{n-2}^*(s), \dots$, for each of the possible states.

CHAPTER 4

DATA COLLECTION AND ANALYSIS

4.0 INTRODUCTION

In this chapter, we shall consider a computational study of dynamic programming for solving product allocation problem.

The choice of the product allocation model is a real life problem in the distribution industry. The aim is to determine the optimal allocation policy in the business so that the business gets the optimum return of profit from the number of salesmen working in different market zone. The general practice is that most establishments do not have a well structured plan on how to allocate salesmen to market zones. Salesmen are allocated by trial and error basis and at the discretion of people or departments in charge. These methods are faulted, and are basically inefficient as returns from salesmen are not optimal.

4.1 DATA COLLECTION AND ANALYSIS

A company has 8 salesmen, who have to be allocated to four marketing zones. The return of profit from each zone depends upon the number of salesmen working that zone. The expected returns for different number of salesmen in different zones, as estimated from the past records, are given in Table 4.1. Determine the optimal allocation policy.

Table 4.1: Sales Marketing in Zones

NO. OF SALESMEN	ZONE 1	ZONE 2	ZONE 3	ZONE 4
0	45	30	35	42
1	58	45	45	54
2	70	60	52	60
3	82	70	64	7
4	93	79	72	82
5	101	90	82	95
6	108	98	93	102
7	113	105	98	110
8	118	110	100	110

All values in Table 4.1 apart from number of salesmen are in thousands (GH¢'000). The problem here is how many salesmen are to be allocated to each zone to maximize the total return. In this problem each zone can be considered as a stage, number of salesmen in each stage as decision variables. Number of salesmen available for allocation at a stage is the state variable of the problem.

In this problem, decision policy requires making four interrelated decisions. What should be the number of salesmen in each of the four marketing zones? If x_1, x_2, x_3 and x_4 are the number of salesmen allocated to the four zones and $f_1(x_1) \dots f_4(x_4)$ are respectively the returns from the four zones, then the objective function is:

Maximize $Z = f_1(x_1) + f_2(x_2) + f_3(x_3) + f_4(x_4).$

Subject to: $x_1 + x_2 + x_3 + x_4 \leq 8$ and x_1, x_2, x_3 and x_4 are non-negative integers.

Thus,

Maximize $Z = \sum_{i=1}^4 f_i(x_i)$

s.t. $\sum_{i=1}^4 x_i \leq 8$

$i \in \mathbb{Z}^+$

Considering the first stage (zone 1) and add to it the second stage (zone 2) and compute the optimal return and optimal allocation. The allocation of salesmen for each zone may be 0, 1, 2 ...and 8.

4.2 TABLES OF RESULTS

Table 4.2: Returns from combination of allocating salesmen to zones 1 and 2

Zone 1 Salesmen →		0	1	2	3	4	5	6	7	8
Return		45	58	70	82	93	101	108	113	118
Zone 2 Salesmen ↓	Return									
0	30	75	88	100	112	123	141	138	143	148
1	45	90	103	115	127	138	146	153	158	
2	60	105	118	130	142	153	161	168		
3	70	115	128	140	152	163	171			
4	79	124	137	149	161	172				
5	90	135	148	160	172					
6	98	143	156	168						
7	105	150	163	—						
8	110	155								

The company can allocate zero salesmen, then zero to zone 1 and zero to zone 2 and the total outcome is 75.

When company wants to allocate 1 salesman to two zones, the allocation is zero to zone 1 and 1 to zone 2 or 1 to zone 1 and zero to zone 2. The outcomes are 90 and 88 respectively.

When company wants to allocate 2 salesmen to two zones, the allocation is 1 to zone 1 and 1 to zone 2 or 0 to zone 1 and 2 to zone 2 or 2 to zone 1 and 0 to zone 2. The outcomes are 103, 100 and 105 respectively.

If the company wants to allocate 3 salesmen to two zones, the allocation is 0 to zone 1 and 3 to zone 2 or 3 to zone 1 and 0 to zone 2 or 2 to zone 1 and 1 to zone 2 or 1 to zone 1 and 2 to zone 2. The outcomes are 112, 115, 118 and 115 respectively.

If the company wants to allocate 4 salesmen to two zones, the allocation is 4 to zone 1 and 0 to zone 2 or 3 to zone 1 and 1 to zone 2 or 2 to zone 1 and 2 to zone 2 or 1 to zone 1 and 3 to zone 2 or 0 to zone 1 and 4 to zone 2. The outcomes are 123, 127, 130, 128 and 124 respectively.

If the company wants to allocate 4 salesmen to two zones, the allocation is 4 to zone 1 and 0 to zone 2 or 3 to zone 1 and 1 to zone 2 or 2 to zone 1 and 2 to zone 2 or 1 to zone 1 and 3 to zone 2 or 0 to zone 1 and 4 to zone 2. The outcomes are 123, 127, 130, 128 and 124 respectively.

If the company wants to allocate 5 salesmen to two zones, the allocation is 5 to zone 1 and 0 to zone 2 or 4 to zone 1 and 1 to zone 2 or 3 to zone 1 and 2 to zone 2 or 2 to zone

1 and 3 to zone 2 or 1 to zone 1 and 4 to zone 2 or 0 to zone 1 and 5 to zone 2. The outcomes are 141, 138, 142, 140, 137 and 135 respectively.

The company can allocate 6 salesmen to two zones, the allocation is 6 to zone 1 and 0 to zone 2 or 5 to zone 1 and 1 to zone 2 or 4 to zone 1 and 2 to zone 2 or 3 to zone 1 and 3 to zone 2 or 2 to zone 1 and 4 to zone 2 or 1 to zone 1 and 5 to zone 2 or 0 to zone 1 and 6 to zone 2. The outcomes are 138, 146, 153, 152, 149, 148 and 143 respectively.

In allocating 7 salesmen to two zones, the allocation is 7 to zone 1 and 0 to zone 2 or 6 to zone 1 and 1 to zone 2 or 5 to zone 1 and 2 to zone 2 or 4 to zone 1 and 3 to zone 2 or 3 to zone 1 and 4 to zone 2 or 2 to zone 1 and 5 to zone 2 or 1 to zone 1 and 6 to zone 2 or 0 to zone 1 and 7 to zone 2. The outcomes are 143, 153, 161, 163, 161, 160, 156 and 150 respectively.

Similarly if the company wants to allocate 8 salesmen to two zones, the allocation is 8 to zone 1 and 0 to zone 2 or 7 to zone 1 and 1 to zone 2 or 6 to zone 1 and 2 to zone 2 or 5 to zone 1 and 3 to zone 2 or 4 to zone 1 and 4 to zone 2 or 3 to zone 1 and 5 to zone 2 or 2 to zone 1 and 6 to zone 2 or 1 to zone 1 and 7 to zone 2 or 0 to zone 1 and 8 to zone 2. The outcomes are 148, 158, 168, 171, 172, 172, 168, 163 and 155 respectively.

Table 4.3: Optimum returns from the allocation of salesmen to zones 1 and 2

No. of Salesmen	0	1	2	3	4	5	6	7	8	
Zone 1	0	0	0	1	2	3	4	4	4	3
Zone 2	0	1	2	2	2	2	2	3	4	5
Outcome	75	90	105	118	130	142	153	162	172	172

KNUST

In the second stage, let us combine zone 3 and zone 4 and get the total market returns.

Table 4.4: Returns from combination of allocating salesmen to zones 3 and 4

Zone 3 Salesmen		0	1	2	3	4	5	6	7	8
Return		35	45	52	64	72	82	93	98	100
Zone 4 Salesmen	Return									
0	42	77	97	94	106	114	124	136	140	142
1	54	89	99	106	118	126	136	147	152	
2	60	95	105	112	124	132	142	153		
3	70	105	115	122	134	142	152			
4	82	117	127	134	146	154				
5	95	130	140	147	159					
6	102	137	147	154						
7	110	145	155							
8	110	145								

If the company wants to allocate zero salesmen, then 0 to zone 3 and 0 to zone 4 and the total outcome is 77.

When company wants to allocate 1 salesman to two zones, the allocation is zero to zone 3 and 1 to zone 4 or 1 to zone 3 and zero to zone 4. The outcomes are 97 and 89 respectively.

When company wants to allocate 2 salesmen to two zones, the allocation is 1 to zone 3 and 1 to zone 4 or 0 to zone 3 and 2 to zone 4 or 2 to zone 3 and 0 to zone 4. The outcomes are 94, 99 and 95 respectively.

If the company wants to allocate 3 salesmen to two zones, the allocation is 0 to zone 3 and 3 to zone 4 or 3 to zone 3 and 0 to zone 4 or 2 to zone 3 and 1 to zone 4 or 1 to zone 3 and 2 to zone 4. The outcomes are 106, 106, 105 and 105 respectively.

If the company wants to allocate 4 salesmen to two zones, the allocation is 4 to zone 3 and 0 to zone 4 or 3 to zone 3 and 1 to zone 4 or 2 to zone 3 and 2 to zone 4 or 1 to zone 3 and 3 to zone 4 or 0 to zone 3 and 4 to zone 4. The outcomes are 114, 118, 112, 115 and 117 respectively.

If the company wants to allocate 4 salesmen to two zones, the allocation is 4 to zone 3 and 0 to zone 4 or 3 to zone 3 and 1 to zone 4 or 2 to zone 3 and 2 to zone 4 or 1 to zone 3 and 3 to zone 4 or 0 to zone 3 and 4 to zone 4. The outcomes are 114, 118, 112, 115 and 117 respectively.

If the company wants to allocate 5 salesmen to two zones, the allocation is 5 to zone 3 and 0 to zone 4 or 4 to zone 3 and 1 to zone 4 or 3 to zone 3 and 2 to zone 4 or 2 to zone 3 and 3 to zone 4 or 1 to zone 3 and 4 to zone 4 or 0 to zone 3 and 5 to zone 4. The outcomes are 124, 126, 124, 122, 127 and 130 respectively.

If the company wants to allocate 6 salesmen to two zones, the allocation is 6 to zone 3 and 0 to zone 4 or 5 to zone 3 and 1 to zone 4 or 4 to zone 3 and 2 to zone 4 or 3 to zone 3 and 3 to zone 4 or 2 to zone 3 and 4 to zone 4 or 1 to zone 3 and 5 to zone 4 or 0 to zone 3 and 6 to zone 4. The outcomes are 136, 136, 132, 134, 134, 140 and 137 respectively.

If the company wants to allocate 7 salesmen to two zones, the allocation is 7 to zone 3 and 0 to zone 4 or 6 to zone 3 and 1 to zone 4 or 5 to zone 3 and 2 to zone 4 or 4 to zone 3 and 3 to zone 4 or 3 to zone 3 and 4 to zone 4 or 2 to zone 3 and 5 to zone 4 or 1 to zone 3 and 6 to zone 4 or 0 to zone 3 and 7 to zone 4. The outcomes are 140, 147, 142, 142, 146, 147, 147 and 145 respectively.

Similarly if the company wants to allocate 8 salesmen to two zones, the allocation is 8 to zone 3 and 0 to zone 4 or 7 to zone 3 and 1 to zone 4 or 6 to zone 3 and 2 to zone 4 or 5 to zone 3 and 3 to zone 4 or 4 to zone 3 and 4 to zone 4 or 3 to zone 3 and 5 to zone 4 or 2 to zone 3 and 6 to zone 4 or 1 to zone 3 and 7 to zone 4 or 0 to zone 3 and 8 to zone 4. The outcomes are 142, 152, 153, 152, 154, 159, 154, 155 and 145 respectively.

Table 4.5: Optimum returns from the allocation of salesmen to zones 3 and 4

No. of Salesmen	0	1	2	3	4	5	6	7	8
Zone 3	0	1	1	2	3	5	1	1	2
Zone 4	0	0	1	1	1	0	5	6	5
Outcome	77	97	99	106	118	130	140	147	159

In third stage we combine both zones 1 and 2 outcomes and zones 3 and 4 outcomes.

Table 4.6: Returns from combination of allocating salesmen to Zones 1 and 2 and zones 3 and 4 combined.

Zone1&2 Salesmen →		(0,0) 0	(0,1) 1	(0,2) 2	(1,2) 3	(2,2) 4	(3,2) 5	(4,2) 6	(4,3) 7	(4,4)(3,5) 8
Return		75	90	105	118	130	142	153	163	172
Zone3&4 Salesmen ↓	Return									
0(0,0)	77	152	167	182	195	207	219	230	240	247
1(1,0)	97	172	187	202	215	227	239	243	260	
2(1,1)	99	174	189	204	217	229	241	252		
3(2,1)	106	181	196	211	224	236	248			
4(3,1)	118	193	208	223	236	248				
5(5,0)	130	205	220	235	248					
6(1,5)	140	215	230	245						
7(1,6)(2,5)	147	222	237							
8(3,5)	159	234								

Table 4.6: Returns from allocating salesmen to Zones 1 and 2 and zones 3 and 4

Salesmen	0	1	2	3	4	5	6	7	8
Zone 1	0	0	1	0	1	2	3	4	4
Zone 2	0	0	0	2	2	2	2	2	3
Zone 3	0	1	1	1	1	1	1	1	1
Zone 4	0	0	0	0	0	0	0	0	0
Returns	152	172	187	202	215	227	239	243	260

The above table shows that how salesmen are allocated to various zones and the optimal outcome for the allocation. Maximum outcome is GH¢260,000.

KNUST



CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

5.0 INTRODUCTION

The aim of this chapter is to give an overall summary of the main concepts presented in this thesis, the use of the dynamic programming for market allocation and distribution problem for a particular company in Ghana.

5.1 FINDINGS AND CONCLUSIONS

In the production and distribution theory under market structure, it is asserted that even though a firm or a company may not be making optimal returns, it should still be operating for certain economic reasons, examples being:

- (i). If it closes down it loses its goodwill created.
- (ii). If it closes down it loses all the customers.
- (iii). With the hope that conditions may improve later, say in the long run for the firm to enjoy optimal return or at least breakeven.
- (iv). To maintain its skilled personnel.

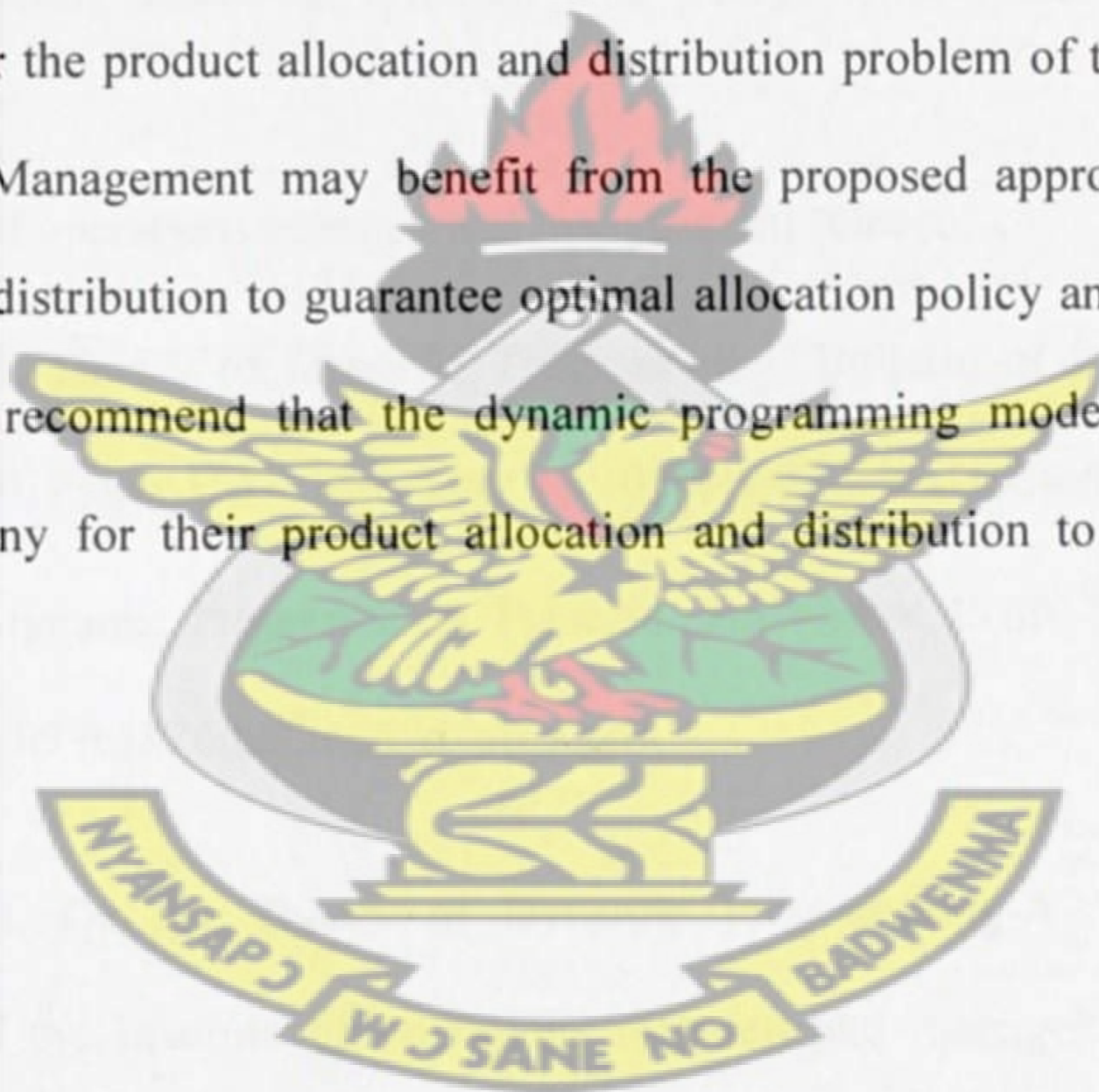
After a thorough examination of the data obtained and analysis made, it was found out that the company carry out its operation based on the above market structure principles.

In this instance, the company from the data given, the optimal allocation policy based on the above principle reach by the company was: allocating 4 salesmen to market zone 1, 2 salesmen to market zone 2, 1 salesman to market zone 3 and 1 salesman to market zone 4, given an optimal return of GH¢252,000.00.

Using our approach, it was observed that the optimal policy that gave maximum achievable value was: allocating 4 salesmen to market zone 1, 3 salesmen to market zone 2, 1 salesman to market zone 3 and 0 salesman to market zone 4, given an optimal return of GH¢260,000.00.

5.2 RECOMMENDATIONS

The use of computer application in computation gives a systematic and transparent solution as compared with an arbitrary method. Using the more scientific dynamic programming model for the product allocation and distribution problem of the company gives a better result. Management may benefit from the proposed approach for the product allocation and distribution to guarantee optimal allocation policy and maximum returns. We therefore recommend that the dynamic programming model should be adopted by the company for their product allocation and distribution to the various market zones.



REFERENCES

1. Alvarez, F. and Stokey, N. (1998). Dynamic programming with homogenous functions. *Journal of Economic Theory*.ISSN: 0022-0531.
2. Archibald, T.W. (2006). Modeling the operation of multireservoir systems using decomposition and stochastic dynamic programming. *Naval Research Logistics: an International Journal*, vol. 53, issue 3, pages 217- 225.
3. Bianco,L., Mingozi, A., and Ricciardelli, S. (1997). Dynamic Programming Strategies for the Traveling Salesman Problem with Time Window and Precedence Constraints.
Journal of the institute of operations research and management science.
4. Bellman, R. (1954). Theory of Dynamic Programming. *Bulletin of the American Mathematical Society* 60. Pages 503 – 506. Doi: 10.109050002-9904-1954-0984-8.
5. Bellman, R. (1957). *Dynamic Programming*. Princeton University, ISBN-486-42809-5.
www.wu-wien.ac.at/usr/h99c1826/bellman_dyprog.pdf.
6. Bellman, R. (1956). On the Theory of Dynamic Programming-A Warehousing Problem. *A journal of the institute of operations research and management science*.
www.mansci.journal.informs.org.
7. Bertsekas, D.P. (2001). *Dynamic Programming and Optimal Control 2nd Edition*, Athena Scientific. ISBN 1-886529-09-4.
8. Bertossi, A.A. and Mei A. (2000).–Constant time dynamic programming on directed reconfigurable networks. <http://ieeexplore.ieee.org/xpl/freeabs>.

9. Charnes, A., and Cooper, W.W. (1955). Generalisation of the warehousing model, O.N.R. Research Memorandum, No.34.

www.mansu.journal.informs.org/content/2/3/272.

10. Cheng-Liang, C. (2003). Solving multi-objective dynamic optimization problems with fuzzy satisfying method. Issue 3.

11. Chia-Wei, C. (2004). Discretized Sub-optimal tracker for nonlinear continuous two-dimensional systems. Asian journal of control, issue 3.

12. Dawen, R.V. (1986). A Note on Positive Dynamic Programming Journal of the institute of operations Research and Management science vol. 11 no. 2 pages 383-384.

13. Deependra, K. (2010). Incorporating Penalty Function to Reduce Spill in Stochastic Dynamic Programming Based Reservoir Operation of Hydropower Plants IEEJ transactions on electrical and electronic engineering. Issue 5, 2010.

14. De Farias D.P. and Van Roy B. (2001). The Linear Programming Approach to Approximate Dynamic Programming. Journal of the institute of operations research and management science. www.or.journal.informs.org

15. Eddy, S.R. (2004). What is Dynamic Programming?: Nature Biotechnology, vol. 22, Pages 909 – 910.

16. Edward, D.T. (2008). Shortest path stochastic control for hybrid electric vehicles international journal of robust and nonlinear control, issue 12.

17. Fang, C.F. (2010). Modeling and simulation of vehicle projection arrival, discharge process in adaptive traffic signal controls. Journal of advanced transportation, Issue 3.

18. Francisco, G.M. (2010). An Equilibrium Theory of Learning, Search, and Wages
econometrica, Issue 2.
19. Frank, S. (2009). Out-of-Core and Dynamic Programming for Data Distribution on
Volume Visualization Cluster. Journal of computer graphics forum.
20. Freire, D.A. (2000). Three-Dimensional Optimization of Urban Drainage Systems
computer-aided civil and infrastructure engineering, Issue 6.
21. Fonnesbeck, C. J. (2005). Solving dynamic wildlife resource optimization
problems using reinforcement learning. Natural resource modeling. Issue 1, 2005.
22. Gerardo, A.C. (2008). Pattern recognition in capillary electrophoresis data using
dynamic programming in the wavelet domain. electrophoresis, Issue 13.
23. Ghahraman, B. (2004). Linear and non-linear optimization models for allocation of a
limited water supply. Irrigation and Drainage, Issue 10.
[www.onlinelibrary.wiley.com/journal/101002/\(ISSN\) 1531-0361/issues](http://www.onlinelibrary.wiley.com/journal/101002/(ISSN)1531-0361/issues).
24. Gregory, G. A. and Warren, P.B. (2000). An Adaptive Dynamic Programming
Algorithm for Dynamic Fleet Management, II: Multiperiod Travel Times. Journal of the
institute of operations research and management science.
25. Guangming, T., Ninghui, S., and Guang R.G.(2008). Improving Performance of
Dynamic Programming via Parallelism and Locality on Multicore Architectures. Journal
of IEEE Transactions on Parallel and Distributed Systems.
26. GroBmann A., Holldobler, S. and Skvortsover O.(2002). Symbolic Dynamic
Prigramming within the Fluent Calculus.
http://www.academicconcepts.net/concepts/222/dynamic_programming.

27. Harilaos, N. (1980). A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem
Journal of the institute of operations research and management science. .
28. Held, M. and Karp R.M. (1961). A dynamic programming approach to sequencing problems. Proceedings of the 16th ACM national meeting, page 71.201-71.204.
29. Jacoboni, I. (2001). Prediction of the transmembrane regions of α -barrel membrane proteins with a neural network-based predictor. Journal of protein science, Issue 4.
30. Johnson, D.S. and Niemi, K. A. (1983) On Knapsacks, Partitions, and a New Dynamic Programming Technique for Trees. Mathematics of Operations Research vol. 8 no.11-14. .
31. Jushan, B. (2003). Computation and analysis of multiple structural change models. Journal of applied econometrics, Issue 1.
32. Kalavathy, S. (2002). Operation Research. Vicas Publication House Pvt. Ltd.
33. Kap, H.K. and Jong W.B. (2000). A Look-Ahead Dispatching Method for Automated Guided Vehicles in Automated Port Container Terminals. Journal of the institute of operations research and management science. www.transci.journal.informs.org.
34. Kossmann, D. and Stocker, K. (2009). Iterative dynamic programming: a new class of query optimization algorithms. Journal ACM Transactions on Database Systems (TODS) Volume 25 Issue 1.
35. Liu, S. (2004). Shape Matching using Dynamic Programming. The Informs Journal of Operations Management.
36. Mahesh, K.S. (2001). Reservoir Operation and evaluation of downstream flow augmentation. Journal of the american water resources association, Issue 3.

37. Manfred, S. (2003). Stochastic optimization for the ruin probability proceedings in applied mathematics & mechanics, Issue 3.
38. Marc, M. A. (2004). Alignment of protein sequences by their profiles. Journals of protein science, Issue 4.
39. Masafumi ,M. (2010). Optimization of Train Speed Profile for Minimum Energy Consumption. IEEJ transactions on electrical and electronic engineering, Issue 3
40. Matthew, M.S, Mateo, R., Henderson, S.G. and Topaloglu, H. (2007). Approximate Dynamic Programming for Ambulance Redeployment. A journal of the institute of Operations Research and Management Science. www.jocjournal.informs.org.
41. Mofya, C.E. and Cole, S.J. (2004). A Dynamic Programming Algorithm for the Generalized Minimum Filter Placement Problem on Tree Structures. Informs Journal on Computing Spring 2009 vol. 21 no. 2, pages 322-332.
42. Mousavi, S.J. and Karamouz, M. (2003). Computational improvement for dynamic programming models by diagnosing infeasible storage combinations Advances in Water Resources Vol. 26, No. 8. pages 851-859. www.citeulike.org/user.
43. Nocedal, J. and Wright S.J. (2006). Numerical Optimisation. Page 9. Springer 2006.
44. Rolfe, T. (2007). Bulletin of the ACMSIG on computer science education, vol. 39, no. 4, pages 54 – 56.
45. Sahid, B. (2010). Exploring the performance of massively multithreaded architectures concurrency and computation: practice & experience, issue 5.
46. Sakoe, H. (1978). Dynamic Programming Algorithm Optimisation for spoken Word Recognition. www.ieeexplore.ieee.org/xpl/login.jsp. doi: 10-1109/TASSP.1978.1163055.

47. Seong-O, S. (2009). Accurate shape from focus based on focus adjustment in optical microscopy. *Microscopy Research and Technique*, issue 5.
www.onlinelibrary.wiley.com/journal/10.1002/ (ISSN) 1097.0029/issue.
48. Smith, K.D. (2005). Dynamic programming and board games: A survey journal of the institute of operations research and management science. www.jor.journal.informs.org.
49. Spjøtvold, J. (2009). Inf, sup control of discontinuous piecewise affine systems, *International journal of robust and nonlinear control*, Issue 13.
50. Steven, L. A. (1975). On Dynamic Programming with Unbounded Rewards a journal of the institute of operations Research and Management science vol. 21 no. 11, Pages 1225-1233.
51. Tsai, J.C. (2005). Flexible and Robust Implementations of Multivariate Adaptive Regression Splines Within a Wastewater Treatment Stochastic Dynamic Program. *Quality and reliability engineering international*, issue 7.
52. Takayuki, S. (2004). Stochastic unit commitment problem. *international transactions in operational research*, Issue 8.
53. Rolfe, T. (1983). Alternative Dynamic Programming Solution for the 0/1 Knapsack *Bulletin of the ACM SIG on Computer Science Education*, Vol. 39, No. 4 , pp. 54-56.
54. Thidarat, T. (2009). ~~Approximate~~ dynamic programming based optimal control applied to an integrated plant with a reactor and a distillation column with recycle. *Aiche Journal*, Issue 3.
55. Tohru, U. (2007). Capacity expansion problems based on fuzzy dynamic programming. [Http://onlinelibrary.wiley.com/doi/10.1002/ecja](http://onlinelibrary.wiley.com/doi/10.1002/ecja).

56. Tracy, J. R. (2006). Optimal eradication: when to stop looking for an invasive plant ecology letters, Issue 7.

57. Warren, B.P. (2009). What you should know about approximate dynamic programming. Naval research logistics: an international journal, Issue 3.

58. Yanhong, L. A. and Scot, D.S. (2002) Program optimization using indexed and recursive data structures. In PEPM '02: Proceedings of the 2002 ACM SIGPLAN workshop on Partial evaluation and semantics-based program manipulation, Vol. 37, Pages 108-118.

59. Yash, P. G. (2001).Solution of nonlinear time-optimal control problems using a semi-exhaustive search. The Canadian Journal of Chemical Engineering, issue 1.

60. Zhang, D.(2009). An Improved Dynamic Programming Decomposition Approach for Network Revenue Management. Vol. 13, No. 1, pages 35 – 52.

<http://msom.journal.informs.org/content/early/2010/10/13/msom.1100.0302>.

