KWAME NKRUMAH UNIVERSITY OF SCIENCE OF

TECHNOLOGY, KUMASI

INSTITUTE OF DISTANCE LEARNING

Application of Dynamic Programming for the Provident Fund Investment Allocation of Ghana Community Network Services Limited Staff (GCNET) Association

A Thesis Submitted to the Institute of Distance Learning, Kwame Nkrumah University of Science and Technology in partial Fulfillment of the Requirements for the Degree of

MSc. INDUSTRIAL MATHEMATICS

Mbiliba Dennis Kojo (BSc Computer Science)

November, 2013

DECLARATION

I hereby declare that this submission is my own work towards the Master of Science degree

and that, to the best of my knowledge it contains no material previously published by another person nor material which has been accepted for award of any other degree of the university except where due acknowledgement has been made in the text.

Mbiliba Dennis Kojo, PG4067810	……..…….	……
Student's Name & ID	Signature	Date

Certified By

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

Certified By

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

ABSTRACT

Fund managers have to decide the amount of a fund's assets and the type of investment Options that should be undertaken, considering the interest rate and the tradeoff between being able to meet shareholder redemptions and minimizing the opportunity cost from lost investment opportunities. In addition, they have to consider redemptions by individuals as well as institutional investors, the current performance of the stock market and interest rates, and the pattern of investments and redemptions which are correlated with market performance. We formulated the provident fund investment problem of Ghana Community Network Services Limited, GCNet, a Ghanaian company as a dynamic program model. The objective is to determine the optimal investment policy in the company so that the business gets the optimum return of profit from the number of investment alternatives. The general practice is that most establishments do not have a well structured plan on how to allocate funds to various investment options in order to maximize returns from the investments. Investment funds 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 the fund invested are not optimal. To overcome this problem, we proposed a dynamic programming algorithm in solving our problem. We use actual data from the funds coffers for market performance and interest rates, and demonstrate the quality of the solution as compared to the existing solution from the firm's investment strategy. We showed from our results that our solution to our investment problem gave an optimal investment returns of GH¢6,600.00, as against the firm's record of investing all the GH¢100 at a return of GH¢6,500.00. The result is a simple policy that describes when money should be moved into and out of cash based on market performance.

ACKNOWLEDGEMENT

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

I am very grateful to my supervisor, Professor S. K. Amponsah of the Department of mathematics, who painstakingly read through every line of the text and offered through his rich experience all the necessary encouragement, direction, guidance, advice and correction for the timely completion of this thesis. I am indeed thankful to him for encouraging the use of accurate grammar and consistent notation in my writings.

I am indebted to Maxwell Kwaku Ofosu of The West African Examination Council (WAEC) for having always been there to listen and give advice. I am deeply grateful to him for the long discussions that helped me sort out the technical details of my work. I am also thankful to him for carefully reading and commenting on countless revisions of this manuscript.

Many friends, especially Franscio Mahama of Ho Polytechnic, Mr. Appiah Wardden Junior and Mr. Owurani Bediako of Ghana Revenue Authority (GRA) Customs Division, Tema, motivated and helped me to persevere through the course. Their support and care encouraged me to overcome setbacks and stay determined on my graduate study. I greatly cherish their friendship and deeply appreciate their belief in me.

I would also like to thank my parents, especially my late father, Jato Mbiliba and mother, Ama Pomaah for their support and encouragement that have brought me this far.

Indeed, none of this would have been possible without the love and patience of my family, especially my wife, Abigail Yajoninga Mbiliba and Children, Tibeitobe Peter Mbiliba and N'Soolimor Paul Mbiliba as well as Auntie Salome of USA.

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

God richly bless you all.

TABLE OF CONTENTS

LIST OF TABLE

CHAPTER 1

OVERVIEW OF CHAPTER 1

This chapter presents an overview of the introduction, the background to the study, problem statement, objectives, methodology, justification, scope of the study, limitations of the study, organisation of the study and summary of the thesis.

1.0 INTRODUCTION

Unlike other management science techniques, dynamic programming does not have a rigid structure or set of equations associated with it. Rather than a technique, it is a procedure for systematically making a set of interrelated decisions that yield an optimum solution. Like linear programming problems, dynamic programming problems have objective functions and constraints, but the form of these functions may vary from problem to problem or even from one part of a problem to the next. The common characteristics of all dynamic programming problems is that they are complex, with many variables, but they can be broken down into several small problems and solved in such a way as to optimize the overall objective function.

Dynamic programming is a useful technique for making a sequence of interrelated decisions. It requires formulating an appropriate recursive relationship for each individual problem. However, it provides a great computational savings over using exhaustive enumeration to find the best combination of decisions, especially for large problems.

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

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. 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. 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"

(R. Bellmand and S. Dreyfus, 1962). 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 ... V_n, with an argument y representing the state of the system at times *i* from 1 to *n*. The definition of $V_n(y)$ is the value obtained in state *y* at the last time *n*. The values V_i at earlier times *i=n-1,n-2,...,2,1* can be found by working backwards, using a recursive relationship called the Bellman equation. For i=2,...n, V_{i-1} at any state y is calculated from V_i by maximizing a simple function (usually the sum) of the gain from decision *i-1* and the function V_i at the new state of the system if this decision is made. Since V_i has already been calculated for the needed states, the above operation yields V_{i-1} for those states. Finally, V_1 at the initial state of the system is the value of the optimal solution. The optimal values of the decision variables can be recovered, one by one, by tracking back the calculations already performed.

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 (Adam B. Levy, 2009).

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. 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. 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.

(*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.

One example of the usefulness of dynamic programming is the resource allocation problems (Murthy C. S. R and Manimaran G. 2001). 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 made three assumptions: (1) the amount of resource assigned to any activity may be any nonnegative value, (2) the benefit obtained from each activity is proportional to the amount of the resource assigned to the activity, and (3) 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 (Gutierrez, 2007), and (Huschka, 2007), equipment replacement problems (Easton, et al., 2003), refinery capacity problems (Murthy, 2001), travelling salesman problem, and Minimax shortest route optimizations (Tomastik, 1993). Thus, dynamic programming has played an important role in supporting managerial decisions in the areas of capital budgeting, warehouse location, and scheduling.

1.2 PROBLEM STATEMENT

This study seeks to apply dynamic programming in solving the provident fund investment problem of Ghana Community Network Services Limited (GCNet) Senior Staff Association. Many optimization problems that arise naturally in applications involve

scenarios where a sequence of interrelated decisions is to be made. Typically, these decisions are to be made over time. In an investment problem, one may need to assemble a new portfolio at the beginning of each day. In general, however, the required decisions may have nothing to do with time. Dynamic programming (DP) is a widely-used

mathematical method for solving linear and nonlinear optimization problems. The term "dynamic" originates from the fact that in most applications, the method is used to derive a sequence of optimal decisions that are adapted to scenario changes that occur dynamically over time.

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.

Formulation. Let the decision variables x_n (n = 1, 2, 3, ..N) be the immediate destination on stage n (the nth stage problem to be solved). Thus, the problem selected is $x_1 \rightarrow x_2 \rightarrow x_3 \ldots \rightarrow x_n$, 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,

$f_n^*(s) = \min f_n(s, x_n) = f_n(s, x_n^*)$,

Where

$f_n(s, x_n)$ = immediate cost (stage n) + minimum future cost (stages n + 1 onward) = c_{sxn} + $f^*_{n+1}(x_n)$.

The value of c_{sxn} 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 $f^*_n(s)$, $f^*_{n-1}(s)$, $f^*_{n-2}(s)$… for each of the possible states s.

1.3 OBJECTIVES

Proper investment decision making is key to success for every investor in their efforts to keep pace with the competitive business environment. Mitigation of exposure to risk plays a vital role, since investors are now directly exposed to the uncertain decision environment. The uncertainty (and risk) of an investment is increasing with the increased number of competing investors entering to market. As a result, the expected return on investment (ROI) of a decision quite often carries a high degree of uncertainty.

Our main objectives are:

(i) To formulate a dynamic programming mathematical model for the investment decision with incorporating this uncertainty in a probabilistic manner,

(ii) Apply policy iteration algorithm of the dynamic programming to solve the model, and

(iii) To select the optimal action (i.e., investment decision) among alternative options for each state to maximize the total gain or reward.

1.4 METHODOLOGY

For our methodology, we propose policy iteration algorithm of the dynamic programming 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 SCOPE OF THE STUDY

As most organizations have embraced and cherished “employee well being”, they devised various plans such as in-service training, welfare contributions, provident fund and many more so as to accomplish their employee wellbeing ambition. This study therefore seeks to mathematically formulate a provident fund investment model which is a real life

problem in most Ghanaian companies for example Ghana Community Network Services Limited (GCNet) Senior Staff Association. The aim is to establish the most favourable investment policy with provident fund management so that the company accrues optimum return of profit from a various investment alternatives for its contributors.

1.7 LIMITATIONS OF THE STUDY

The study is limited to provident fund investment management using dynamic programming approach. This is a deliberate effort on the researcher's part to make the study manageable given the time and resources available to the researcher to complete the study. The study was limited to the perceived effect of application of dynamic programming on an institutional staff provident fund management of Ghana Community Network Services Limited (GCNet) Senior Staff Association.

1.8 ORGANIZATION OF THE STUDY

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.

1.9 SUMMARY

In this chapter, we considered the background, the problem structure and objective of the study - the justification, scope and limitations of the study were also put forward. The next chapter presents relevant literature on the application of dynamic programming.

CHAPTER 2

LITERATURE REVIEW

At present, investment decision making is a critical task because every investment exhibits at least some amount of risk and uncertainty. These risks and uncertainties are the results of huge business competition and vibrant market economy. As a result, recent research in investment decision making is undergoing a paradigm shift with much integration of new techniques with existing methods to develop robust decision making processes, Heikkinen et al., (2009).

Net present value (NPV) is the most common method in investment evaluation, Wang (1998). Alkaraan and Northcott (2006) analyzed the use of conventional investment appraisal techniques such as payback, return on assets (ROA), return on investment (ROI), internal rate of return (IRR), NPV and risk analysis approaches such as sensitivity analysis, adjustment of the payback period, or discount rate.

Handling risk and uncertainty in projects is currently one of the main topics of interest for researchers and practitioners working in the area of project investment decision making Raz and Michael (2001), Topaloglou et al., (2002) and Rockafellar et al., (2000) solved Portfolio optimization problems with dynamic programming and they used Monte Carlo simulations to capture the risk in each associated investment.

Xu-song and Jian-mou (2002) studied the investment decision-making of a project with deterministic dynamic programming.

Yan and Bai (2009) formulated a deterministic dynamic programming model to allocate funds between stocks in a portfolio to maximize income. They captured the risk issues by incorporating the positive correlation between risks and returns of a stock to a large extent.

Heikkinen and Pietola (2009) studied the use of stochastic programming approaches to make optimal investment decisions by modeling the problem as a Markov decision process. A dynamic uncertainty cost is presented with the modification of the classical expected value of perfect information to a dynamic setting. Dixit and Pindyck (1994) described the use of a Markov decision process (MDP) defined in continuous time and with a continuous state space for optimal investment decisions.

Most of the prior research does not consider the inter-related dynamics of the systems that can be encountered by a stochastic dynamic investment model, Botterud (2007).The present research herein aims to handle this uncertainty and interrelated dynamics using infinite horizon stochastic dynamic programming and at the same time make optimal investment decisions for which the maximum total expected reward can be achieved.

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 was achieved so

that the computer time for solving the model was reduced more than 50 times for the reservoir system under study. This methodology was applied to a four-reservoir system located in Iran.

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

Alvarez et al. (1998) presented a model that used dynamic programming in resolving economic issues, particularly homogenous problems. This was made possible by defining the basic existence, uniqueness and convergence results generated from dynamic programming methods. The authors approach provided 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 provided 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 the study was 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 straight-forward 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. David (2005) presented a survey of those problems 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

Hiroaki (1978) studied an optimum dynamic programming (DP) based time-normalization algorithm for spoken word recognition. First, a general principle of time-normalization was given using time warping functioning. Then, two time-normalized distance definitions, symmetric and asymmetric forms, were derived from the principle. These two forms were compared with each other through theoretical discussions and experimental studies. The symmetric form algorithm superiority was established. A new technique, called slope constraint, was successfully introduced, in which the warping function slope was restricted so as to improve discrimination between words in different categories, and 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 the author's paper involves the relative superiority of either a symmetric form of DP-matching or an asymmetric one. In the asymmetric form, time-normalization was achieved by trans-forming the time axis of a speech pattern onto that of the other. In the symmetric form, on the other hand, both time axes were transformed onto a temporarily defined common axis. Theoretical and experimental

comparisons showed that the symmetric form gave better recognition than the asymmetric one.

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. The authors proposed a simultaneous dynamic programming approach to solve the resulting problem, which is a nonlinear optimization problem with nonlinear constraints. The authors showed that their approximation lead to a tighter upper bound on optimal expected revenue than some known bounds in the literature. Their 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 was only slightly higher than the classical version. A numerical study showed 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 was 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. The authors tested and evaluated their method on a database of 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 the authors called Iterative Dynamic Programming or IDP for short. IDP has several important advantages: First, IDP-algorithms produced the best plans of all known algorithms in situations in which dynamic programming are 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 (DPA) 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 to 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 this new algorithm can figure out the optimal solution with a speed 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, the authors developed an optimal polynomial-time dynamic programming algorithm for this problem. The authors compared the performance of the dynamic programming algorithm against a mixed-integer programming model on randomly generated tree networks at varied levels of security and concluded that their algorithm outperformed that of the mixed-integer programming model.

Matthew et al., (2007) presented an approximate dynamic programming approach for making ambulance redeployment decisions in an emergency medical service system. The primary decision was to redeploy idle ambulances so as to maximize the number of calls reached within a delay threshold. The authors began by formulating this problem as a dynamic program. To deal with the high-dimensional and uncountable state space in the dynamic program, they constructed approximations to the value function that were parameterized by a small number of parameters. The authors tune the parameters using simulated cost trajectories of the system. Computational experiments demonstrated the

performance of the approach on emergency medical service systems in two metropolitan areas. They reported 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 was difficult to exploit either parallelism or locality. Worse still, the emerging multi/many-core architectures with small on-chip memory made these issues more challenging. In their paper, the authors addressed the challenges of exploiting the fine grain parallelism and locality of NPDP on multicore architectures. They described 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, they proposed 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, they formulated 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 maps it to the NP-complete precedence constrained job-shop scheduling problem. The authors introduced a k-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. The authors slab-projection slice, introduced, tracks empty space across consecutive slices of data to reduce the amount of data distributed and rendered. It was used to facilitate out-of-core bricking and k-tree partitioning. Load balancing using each of our approaches was 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 the aim to reduce the amount of spill during operation of the reservoir. SDP models with various inflow process assumptions were 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 was 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 were 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 were plotted for a study horizon of 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 was marginally reduced.

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 addressed 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 25 to 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). According to the authors, their 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.

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 the search stops 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. The authors characterized this process and analyzed its interactions with job creation and wage determination. Their 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, their analysis provided a set of useful results on dynamic programming with optimal learning.

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 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.

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 (GM) 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 was also mentioned.

Spjotvold (2009) considered the worst-case optimal control of discontinuous piecewise affine (PWA) systems, which were 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, the authors proposed a method that is guaranteed to obtain a sub-optimal solution, and where the degree of sub-optimality can be specified a priori. This was 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. The authors incorporated 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 (45,km 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 was concluded that the queue length and queue storage ratio are the most appropriate performance measures in real-time signal control of interchanges.

Jushan Bai (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.

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 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 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 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 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. We propose 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 basic functions Mmax, 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 Mmax, 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 × 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 Chen (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: (1) the 2-D optimal-linearization approach of the nonlinear 2-D Roesser's Model (RM), (2) the dynamic programming-based discretized quadratic optimal tracker for linear continuous 2-D systems, (3) the steady-state discretized quadratic sub-optimal tracker for linear continuous 2-D systems, and (4) 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 in [3], 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 enumerating 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 Van (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 queueing 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)×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 ν ε V to all other nodes can be found in constant time by an $O(n^2w)$ x $O(n^2w)$ 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.

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 (TFN). Multiplication or division of TFN does not give rigid TFN, but we approximate them to TFN 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. We 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 optional 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, Chengiun 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 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 Pptimization 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 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 f 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 recomputing 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.

Proper investment decision making is key to success for every investor in their efforts to keep pace with the competitive business environment. Mitigation of exposure to risk plays a vital role, since investors are now directly exposed to the uncertain decision environment. The uncertainty (and risk) of an investment is increasing with the increased number of competing investors entering to market. As a result, the expected return on

investment (ROI) of a decision quite often carries a high degree of uncertainty. Noor and Doucette (2009) studied a model which objective was to formulate a dynamic programming mathematical model for the investment decision with incorporating this uncertainty in a probabilistic manner. Policy iteration algorithm of the dynamic programming was adopted to solve the model. The authors simulation result showed that the algorithm was able to help in taking optimum investment decision.

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 – we shall first put forward the organizational of the company under discussion.

3.1.0 PROFILE OF THE COMPANY

The Ghana Community Network Services Limited (GCNet) in Ghana in December 2003 to facilitate a fast and effective processing of cargo clearance related operations. The GCNet system consists of two main components:

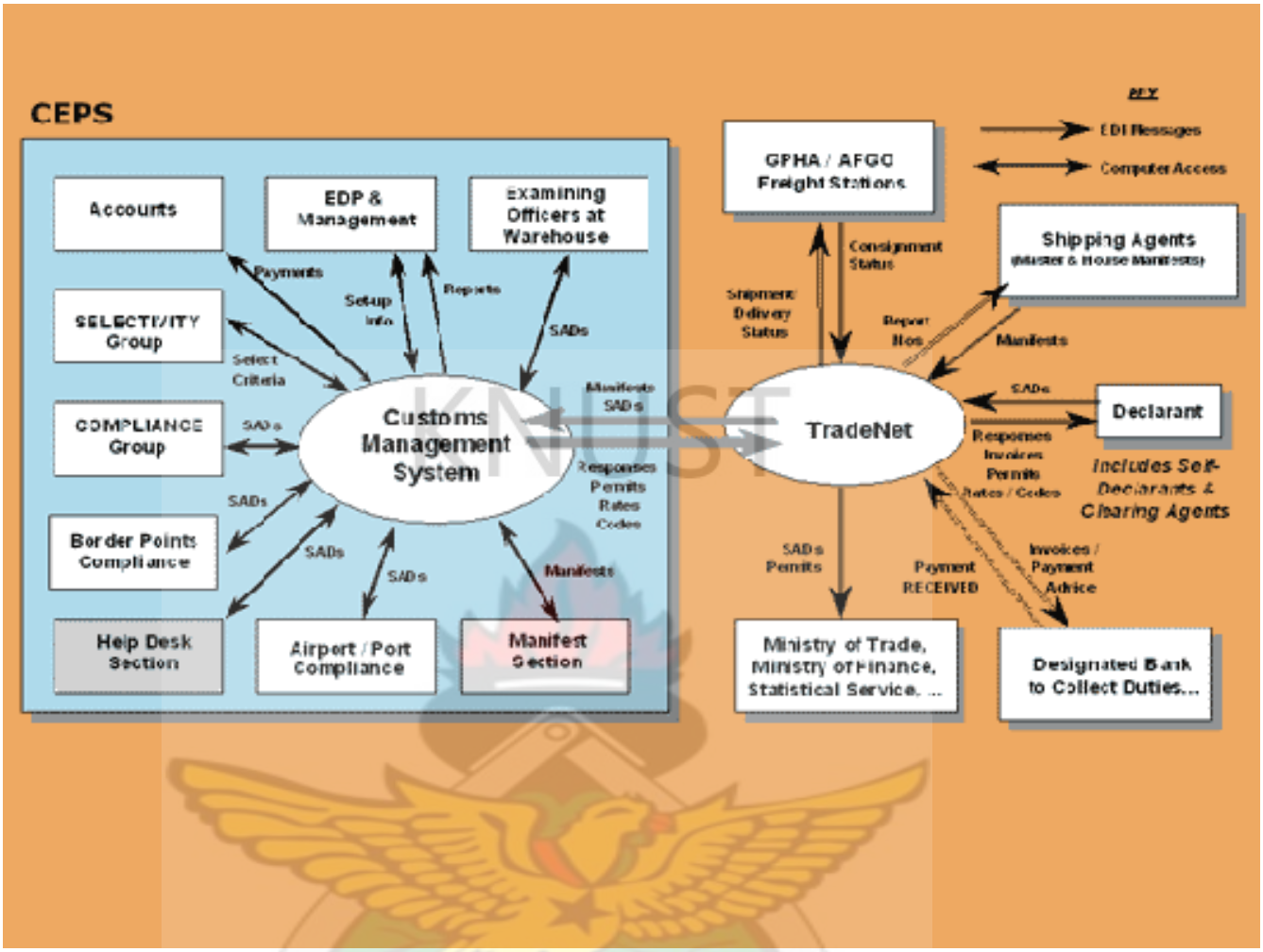
3.1.1 Ghana Customs Management System (GCMS)

This is intended to provide the Ghana Customs, Excise and Preventive Service with a complete integrated computerised system for the processing and management of Customs Declarations and related activities. This system is designed to work in an EDI (Electronic Data Interchange) environment in which Manifests and Customs Declarations are electronically received and automatically processed.

3.1.2 Ghana TradeNet

It is a platform which enables GCMS to share data with the various parties involved in the processing of trade documents and customs clearances.

The illustration below indicates how the system is designed:



GCNet is designed to cater for the following activities:

- Electronic submission of Master Manifests by Shipping Agents and Airlines.
- Electronic submission of House Manifests by Forwarding Agents / Consolidators.
- Electronic submission of Customs Declarations by Clearing Agents or Self-Declarants.
- Payment of Duties & Taxes confirmed electronically by the banks.
- Electronic transmission of Customs clearance approvals to Terminal Operators
- Electronic transmission of Delivery Orders from Shipping Agents to Terminal Operators
- Integrated system for the sharing of files between Customs Officers.

- Transfer of electronic messages between Customs, Traders and other parties concerned.
- Import, Export, Warehousing, Free Zone and Transit operations.

GCNet offers major benefits to Customs and to the Trading Community as a whole:

- Submission of Declarations 24 hours a day, 7 days a week.
- Validation of Declarations performed automatically by the system.
- Front-End Software for Declarants enabling internal statistics.
- Integrated Risk Assessment Module for Customs.
- Payment of Duties and Taxes may be effected at any of the participating banks
- A powerful monitoring tool for Customs.
- An integrated database enabling accurate trade statistics.

3.2 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.

1. 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.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

3.3 The 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.

3.4 SUMMARY

This chapter discussed the researched methodology of the study. In our introduction, we discussed the characteristics of dynamic programming problems. The dynamic programming algorithm was also presented.

The next chapter presents the data collection and analysis of the study.

CHAPTER 4

DATA COLLECTION AND ANALYSIS

4.0 INTRODUCTION

In this chapter, we shall consider a computational study of dynamic programming for solving provident fund investment problem of Ghana Community Network Services (GCNet) Limited Senior Staff Association. The choice of the provident fund investment model is a real life problem in Ghanaian company. The aim is to determine the optimal investment policy in the company so that the business gets the optimum return of profit from the number of investment alternatives. The general practice is that most establishments do not have a well structured plan on how to allocate funds to various investment options in order to maximize returns from the investments. Investment funds 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 the fund invested are not optimal.

4.1 Data Collection and Analysis

The senior staff association of Ghana Community Network Services (GCNet) Limited, operates a provident fund and the management of the fund is considering investing an amount of GH¢100, 000 for the next financial year and that there are four alternatives available.

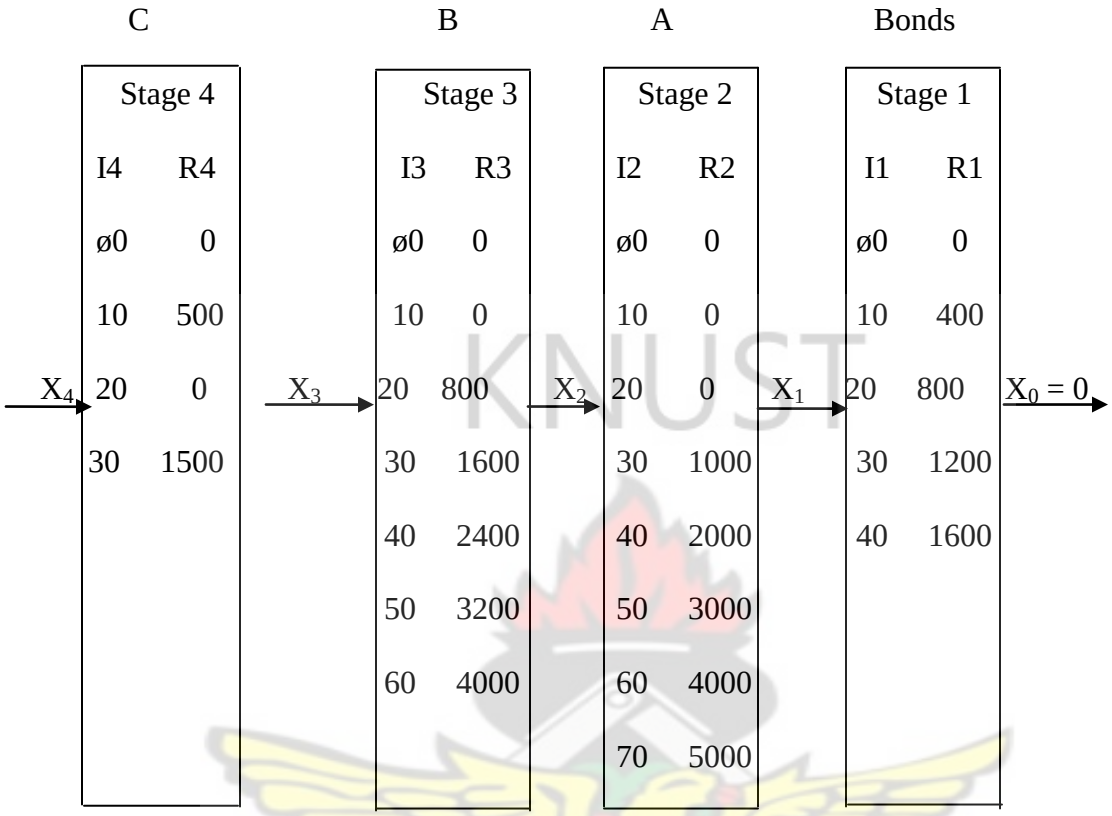
1. Only GH¢40, 000 or less may be invested in corporate bonds that will return 4%.

2. Venture A may be financed at various levels and will return 10% on all investments above the GH¢20, 000 machinery lease. No more than GH¢70, 000 may be invested in venture A.
3. Venture B may be financed at various levels and will return 8% on all investment above the GH¢10, 000 franchise fee. Investment in venture B may not exceed GH¢60, 000.
4. Investment C may be financed at various levels and will return 0.05%. Investment C may not exceed GH¢30, 000.

All GH¢100,000 must be invested, and decisions are restricted to GH¢10, 000 increments. The problem at hand is to determine the optimal investment policy that will generate an optimum total return on investments.

In this problem each investment option can be considered as a **stage**, amount in each stage as **decision** variables. The amount available for investment at a stage is the **state variable** of the problem.

Figure 4.1 shows the investment opportunities as well as the returns



If R_i represents the return from stage I, X_i represents available funds passed from stage to stage, and I_i represents the amount invested in alternative I, the objective is to maximize total return (TR) for stage I and all subsequent stages

$$Tr_i^* = Max \{Rj + Tr_{i-1}^*\}$$

The transition function is

$$X_{i-1} = X_i - I_i$$

Thus, in words, the amount passed out of a stage is equal to the amount passed in less the amount invested.

Stage 1: The stage 1 calculations are presented in Table 4.2. The limits of X_1 and I_1 are both GH¢0 minimum and GH¢40, maximum. X_1 may be as little as GH¢0 because it is possible to exhaust the entire GH¢100, 000 in the other three stages. I_1 may be GH¢0

because we are not required to invest in bonds. There is only one value of I_1 for each value of X_1 because the same amount must be invested as is passed in.

Table 4.2 An Investment calculation for stage 1 (Investment in Bonds)

Available funds for Investment (X_1)	Amount Invested (I_1^*)	Return on Investment R_1^*	X_0^*	Total Return
0	0	0	0	0
10	10	400	0	400
20	20	800	0	800
30	30	1200	0	1200
40	40	1600	0	1600

Stage 2: The stage 2 calculations are presented in Table 4.3. In stage 2, I_2 may range from GH¢0 to GH¢70, 000, and X_2 may range between 10, 000 and 100, 000. Ten thousand cedis is the amount that would be passed if the maximum amount were absorbed in stage 3 and 4, and 100,000 would be passed if none were absorbed in stages 3 and 4. There are no entries for $X_2 = 20$ and $I_2 = 30$, because the amount invested cannot exceed the amount passed in. The same holds true for $X_2 = 30$, $I_2 = 40$, and on down the table to $X_2 = 60$, $I_2 = 70$. There are no entries from $X_2 = 50$, $I_2 = 0$ to $X_2 = 100$, $I_2 = 50$, because X_1 may not exceed 40 and $X_1 = X_2 – I_2$. Thus $X_2 – I_2$ may not exceed 40.

Table 4.3 An Investment calculation for stage 2 (Investment in Venture A)

Available funds for Investment (X_2)	Amount Invested (I_2^*)	Return on Investment R_2^*	X_1^*	Total Return
10	0	0	10	400
	10	0	0	0
20	0	0	20	800
	10	0	10	400
	20	0	0	0
30	0	0	30	1200
	10	0	20	800
	20	0	10	400
	30	1000	0	1000
40	0	0	40	1600
	10	0	30	1200
	20	0	20	800
	30	1000	10	1400
	40	2000	0	2000
50	10	0	40	1600
	20	0	30	1200
	30	1000	20	1800
	40	2000	10	2400
	50	3000	0	3000

60	20	0	40	1600
	30	1000	30	2200
	40	2000	20	2800
	50	3000	10	3400
	60	4000	0	4000
70	30	1000	40	2600
	40	2000	30	3200
	50	3000	20	3800
	60	4000	10	4400
	70	5000	0	5000
80	40	2000	40	3600
	50	3000	30	4200
	60	4000	20	4800
	70	5000	10	5400
90	50	3000	40	4600
	60	4000	30	5200
	70	5000	20	5800
100	60	4000	40	5600
	70	5000	30	6200

Stage 3: stage 3 calculations are similar to that of sage 2 calculations and are shown in Table 4.4.

Table 4.4 An Investment calculation for stage 3 (Investment in Venture B)

Available funds for Investment (X_3)	Amount Invested (I_3^*)	Return on Investment R_3^*	X_2^*	Total Return
70	0	0	70	5000
	10	0	60	4000
	20	800	50	3800
	30	1600	40	3600
	40	2400	30	3600
	50	3200	20	4000
	60	4000	10	4400
80	0	0	80	5400
	10	0	70	5000
	20	800	60	4800
	30	1600	50	4600
	40	2400	40	4400
	50	3200	30	4400
	60	2400	20	4800
90	0	0	90	5800
	10	0	80	5400
	20	800	70	5800
	30	1600	60	5600

	40	2400	50	5400
	50	3200	40	5200
	60	4000	30	5200
100	0	0	100	6200
	10	0	90	5800
	20	800	80	6200
	30	1600	70	6600
	40	2400	60	6400
	50	3200	50	6200
	60	4000	40	6000

Stage 4: The stage 4 calculations are presented in Table 4.5. When the returns for all feasible combinations of X_i and I_i have been calculated for each table, determine the optimum value and record it in the R^* column. In the I_* column record the I value for which the optimum R^* value occurs.

Table 4.5 An Investment calculation for stage 4 (Investment in Venture C)

Available funds for Investment (X_4)	Amount Invested (I_4^*)	Return on Investment R_4^*	X_3^*	Total Return
100	0	0	100	6600
	10	500	90	6300
	20	1000	80	6400
	30	1600	70	6500

After completing the tables of stages, we begin with the last table completed and work back through the tables to identify the overall optimum strategy, shown in Table 4.6

Table 4.6 Solution to our Investment Problem

Stage	Xi*	Ii*	Si-1*	R*
4	100	0	100	0
3	100	30	7	1600
2	70	70	0	5000
1	0	0	0	0
Total Return:				6,600

4.2 Results

From the summarized Table 4.6 above, which shows the solution to our investment problem, it could be seen that the optimal investment policy is revealed in stages three and two, with respective returns of GH¢1,600.00 by investing in GH¢30 and GH¢5,000.00 by investing in GH¢70, given a total return of GH¢6,600.00, as against the firm's record of investing all the GH¢100 at a return of GH¢6,500.00.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.0 INTRODUCTION

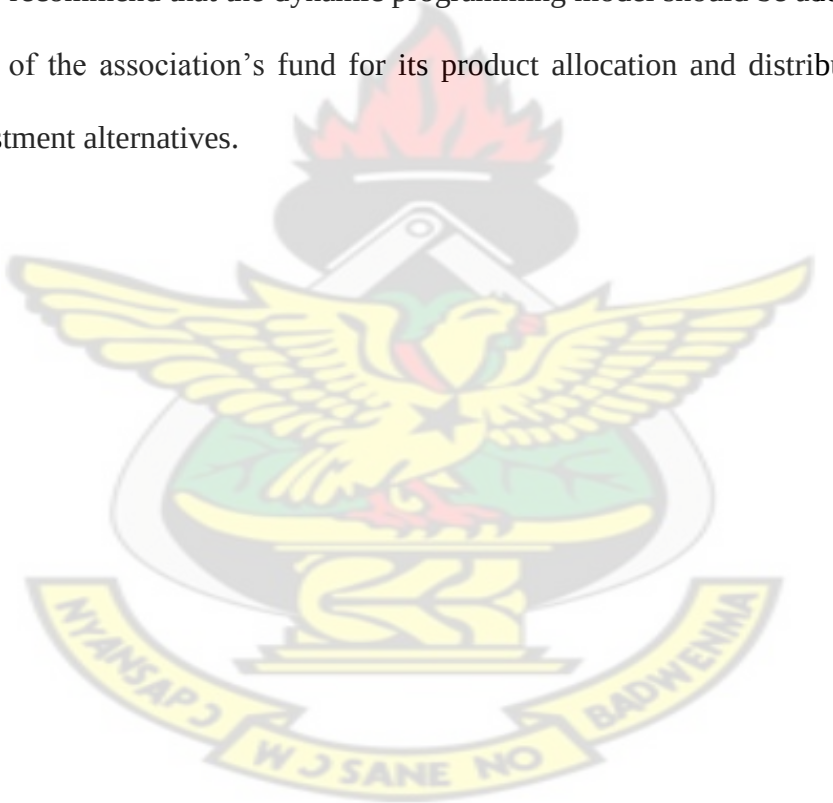
The aim of this chapter is to give an overall summary of the main concepts presented in these studies, the use of the dynamic programming for provident fund investment allocation and distribution problem for Ghana Community Network Services (GCNet) Limited Senior Staff Association.

5.1 FINDINGS AND CONCLUSIONS

The fund managers have to decide the amount of fund's assets that should be kept in cash, considering the tradeoff between being able to meet shareholder redemptions and minimizing the opportunity cost from lost investment opportunities. In additions, they have to consider redemptions by individuals as well as institutional investors, the current performance of the stock market and interest rates, and the pattern of investment and redemptions which are correlated with market performance. We formulated the problem as a dynamic program. We also adapted the dynamic programming algorithm to solve the problem. We use actual data for market performance and interest rates, and demonstrated the quality of the solution for the four alternatives investments. Our results show that the optimal solution obtained outperforms that of the existing model. The result is a simple policy that describes how much money should be moved into and out of cash based on market performance.

5.2 RECOMMENDATIONS

The use of mathematical model in computer application gives a systematic and transparent solution as compared with an arbitrary method. Using the more scientific dynamic programming model for the provident fund allocation and distribution problem of the Ghana Community Network Services (GCNet) Senior Staff Association gives a better result. Management may benefit from the proposed approach for the investment allocation and distribution to guarantee optimal allocation policy and maximum returns. We therefore recommend that the dynamic programming model should be adopted by the management of the association's fund for its product allocation and distribution to the various investment alternatives.



REFERENCE

1. AGroBmann Holldobler S and Skvortsover O (2002). Symbolic Dynamic Prigramming within the fluent calculus. Academicconcepts.net/concepts/222/dynamic_programming.

2. Alvarez F. and Stokey N. (1998). Dynamic programming with homogenous functions Journal of Economic Theory ISSN: 0022-0531.

3. Archibald T. W. (2006). Modeling the operation of multireservoir systems using decomposition and stochastic dynamic programming**.** NAVAL RESEARCH LOGISTICS: AN INTERNATIONAL JOURNAL**,** Issue 3.

4. Aristide M., Lucio B. and Salvatore R. (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.

5. Bertossi A. A. and Mei A. (2000). Constant time dynamic programming on directed reconfigurable networks. http://ieeexplore.ieee.org/xpl/freeabs.

6. Bijan G. (2004). Linear and non-linear optimization models for allocation of a limited water supply. Irrigation and Drainage, Issue.

7. Cheng-Liang C. (2003). Solving multi-objective dynamic optimization problems with fuzzy satisfying method. OPTIMAL CONTROL APPLICATIONS AND METHODS, Issue 5.

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

9. Chisonge M. 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. 2 322-332.

10. Fonnesbeck C. J. (2005).Solving dynamic wildlife resource optimization problems using reinforcement learning. Natural resource modeling, Issue 1.

11. Zhang D. (2009). An Improved Dynamic Programming Decomposition Approach for Network Revenue Management. Transportation Science. The Informs Journal of Operations Management.

12. David K. S. (2005). Dynamic programming and board games: A survey. Journal of the institute of operations research and management science.

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.

14. DeFarias D. P. and Van R. B. (2001). The Linear Programming Approach to Approximate Dynamic Programming. Journal of the institute of operations research and management science.

15. Kossmann D. and Konrad S. (2009). Iterative dynamic programming: a new class of query optimization algorithms. Journal ACM Transactions on Database Systems (TODS) Volume 25 Issue 1.

16. Tate E. D. J. (2008). Shortest path stochastic control for hybrid electric vehicles international journal of robust and nonlinear control.

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. Gonzalez F. 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 A. D. (2000). Three-Dimensional Optimization of Urban Drainage Systems computer-aided civil and infrastructure engineering, Issue 6.

21. Ceballos G. A. (2008). Pattern recognition in capillary electrophoresis data using dynamic programming in the wavelet domain. Electrophoresis, Issue 13.

22. Gregory A. G. and Warren B. P. (2000). An Adaptive Dynamic Programming Algorithm for Dynamic Fleet Management, II: Multiperiod Travel Times. Journal of the institute of operations research and management science.

23. Guang R. G., Guangming T. and Ninghui S. (2008). Improving Performance of Dynamic Programming via Parallelism and Locality on Multicore Architectures. Journal of IEEE Transactions on Parallel and Distributed Systems.

24. Psaraftis H. 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.

25. 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.

26. Hiroaki S. (1978). Dynamic programming algorithm optimization for spoken word recognition. citeseerx.ist.psu.edu/viewdoc/summary.

27. Topalogl H., Matthew S. M., Restrepo M, and Shane G. H. (2007). Approximate Dynamic Programming for Ambulance Redeployment. A journal of the institute of Operations Research and Management Science.

28. Jacoboni I. (2001). Prediction of the transmembrane regions of ,-barrel membrane proteins with a neural network-based predictor. Journal of protein science**,** Issue 4.

29. 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. 1 114.

30. Tsai J. C. 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.

31. Jushan B. (2003). Computation and analysis of multiple structural change models. Journal of applied econometrics, Issue 1.

32. Kap Hwan 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

33. Karamouz M and Mousavi S J (2003). Computational improvement for dynamic programming models by diagnosing infeasible storage combinations Advances in Water Resources Vol. 26, No. 8. pp. 851-859.

34. Liu S (2004). Shape Matching using Dynamic Programming.The Informs Journal of Operations Management.

35. Mahesh K. S. (2001). Reservoir Operation ani evaluation of downstream flow augmentation**.** Journal of the american water resources association, Issue 3.

36. Manfred S. (2003). Stochastic optimization for the ruin probability proceedings in applied mathematics and mechanics**,** Issue 3.

37. Marti-Renom M. A. (2004). Alignment of protein sequences by their profiles.

Journals of protein science, Issue 4.

38. Masafumi M. (2010). Optimization of Train Speed Profile for Minimum Energy Consumption. ieej transactions on electrical and electronic engineering.

39. Rein L. (2001). Further developments in the new approach to boundary condition iteration in optimal control. the canadian journal of chemical engineering, Issue 6.

40. Bellman R. (1956). On the Theory of Dynamic Programming-A Warehousing Problem. A journal of the institute of operations research and management science.

41. Rolf V. D. (1986). A Note on Positive Dynamic Programming Journal of the institute of operations Research and Management science vol. 11 no. 2 383-384.

42. Sahid B. (2010). Exploring the performance of massively multithreaded architectures concurrency and computation: practice & experience, issue 5.

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

44. Seong-O S. (2009). Accurate shape from focus based on focus adjustment in optical microscopy., Issue 5.

45. ShiHai D, YiSong C. and GuoPing W. (2003). Further improvement on dynamic programming for optimal bit allocation. Journal of Computer Science and Technology.

46. Spjøtvold J. (2009). Inf,sup control of discontinuous piecewise affine systems

International journal of robust and nonlinear control, Issue 13.

47. Lippman S. A. (1975). On Dynamic Programming with Unbounded Rewards a journal of the institute of operations Research and Management science vol. 21 no. 11 1225-1233.

48. Takayuki S. (2004). Stochastic unit commitment problem. international transactions in operational research.

49. 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.

50. Rolfean T. J. (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.

51. Tohru U. (2007). Capacity expansion problems based on fuzzy dynamic programming. onlinelibrary.wiley.com/doi/10.1002/ecja.

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

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

54. Yash Bang-Bang G. P (2001). Solution of nonlinear time-optimal control problems using a semi-exhaustive search. THE CANADIAN JOURNAL OF chemical engineering, Issue 1.

55. http://www.gcnet.com.gh/procedures/procedure.asp

56. Adam B. L (2009). The Basics of Practical Optimization http://books.google.com.gh/books

57. Bellman R. and Dreyfus S. (1962) Applied Dynamic Programming. Princeton University Press Princeton, New Jessey.

58. Heikkinen T., Pietola K (2009) Investment and the dynamic cost of income uncertainty: European Journal of Operational Research, - Elsevier

59. Wang A. J. (1998). Key concepts in evaluating outcomes of ATP funding of medical technologies. The Journal of Technology Transfer.

60. Alkaraan F, Northcott D (2006). Strategic capital investment decision-making: A role for emergent analysis tools? A study of practice in large UK manufacturing companies. The British Accounting Review.

61. Raz T, Michael E (2001). Use and benefits of tools for project risk management International Journal of Project Management.

62. Topaloglou N, Vladimirou H, Zenios S. A. (200). CVaR models with selective hedging for international asset allocation. Journal of Banking & Finance.

63. Rockafellar R. T, Uryasev S. (2000) Optimization of conditional value-at-risk Journal of risk.

64. Xu-song X, Jian-mou W. (2002). A dynamic programming algorithm on Project-Gang investment decision-making. Wuhan University Journal of Natural Sciences.

65. Murthy C. S. R, Manimaran G (2001). Resource management in real time systems and networks. books.google.com

65. Mousavi S. J, Karamouz M (2003). Computational improvement for dynamic programming models by diagnosing infeasible storage combinations

Advances in water resources.

66. Alvarez F, Stokey N. L. (1998). Dynamic programming with homogeneous functions. Journal of economic theory.