DYNAMIC PROGRAMMING BASED BUS REPLACEMENT POLICY FOR METRO-MASS TRANSIT LIMITED-KUMASI DEPOT

ABDUL RASHID (BSc Computer Science)

A thesis submitted to the Department of Mathematics, Kwame Nkrumah University of Science and Technology, Kumasi in partial fulfilment of the requirements for the degree of

Master of Science

Industrial Mathematics

OCTOBER, 2011

DECLARATION

I hereby declare that this submission is my own work towards the Master of Science Industrial Mathematics and that, to the best of my knowledge it contains no material previously published by another person or material which has been accepted for award of any other degree of the university except where due acknowledgement has been made in the text.

Abdul Rashid, PG3005509	………………….	……………………
Student's Name & ID	Signature	Date

Certified by:

Mr. F. K. Darkwah	………………….	……………………
Supervisor's Name	Signature	Date

Certified by:

Prof. I.K Dontwi	………………….	……………………
Dean of IDL	Signature	Date

Certified by:

Mr. F. K. Darkwah	………………….	……………………
Head of Department's Name	Signature	Date

ABSTRACT

Recent advances in automobile replacement have made possible the deployment of several strategies aimed at minimising the total operational cost on one hand and maximising the total net profit on the other hand of an equipment in service of a given organization.

This thesis looks at the various operational costs associated with running an automobile bus as a background to making a future replace or keep decision throughout a given planned horizon. Our work focuses on using dynamic programming in solving the backward profit recursive relation for an optimal replacement policy via Microsoft Excel solver implementation.

This approach finds the optimal replacement policies to be replaced in every five and four years of the bus's service life: Verband Deutscher Lokomotivindustrie (VDL) Commuter, Verband Deutscher Lokomotivindustrie (VDL) Neoplan City (2^{nd} generation), Verband Deutscher Lokomotivindustrie (VDL) Daf and Verband Deutscher Lokomotivindustrie (VDL)Jonckheere buses should be replaced every five (5) years with Neoplan City (1^{st} generation) bus replaced every four (4) years.

TABLE OF CONTENT

Content *Page*

Declaration ii

Abstract.. iii

Table of content iv

List of tables vii

List of figures ix

Dedication x

Acknowledgement xi

CHAPTER ONE: INTRODUCTION

1.1 Background to the study………………………………………………... 1

1.2 Public transport in Ghana………………………………………………... 2

1.2.1 Background to Metro Mass Transit Limited………………………… 3

1.3 Description of bus maintenance.. .. 4

1.4 Problem statement………………………………………………………… 5

1.5 Objective of the study……………………………………………………… 6

1.6 Methodology……………………………………………………………….. 7

1.7 Justification……………………………………………………………….. 7

1.8 Thesis organization………………………………………………………… 8

CHAPTER TWO: LITERAURE REVIEW

2.1 Dynamic programming review... 9

2.2 Replacement problems review... 11

CHAPTER THREE: RESEARCH METHODS

3.1 Optimization techniques... 18

3.2 Recursive programming.. 19

3.2.1 Divide-and-conquer algorithm... 19

3.2.2 Greedy algorithm.. 22

3.3 Principles of dynamic programming………………………………………… 25

3.3.1 Dynamic programming applications.. 27

3.3.2 Dynamic programming strengths.. 28

3.3.3 Dynamic programming shortcomings............................... 29

3.3.4 Comparative advantage... 30

3.4 A production and inventory control problem....................................... 31

3.5 The stagecoach problem (network problem).. 44

3.6 The knapsack problem.. 53

3.7 The equipment-replacement problem with trade-in cost............................. 61

3.8 Automobile replacement problem with or without income........................... 69

CHAPTER FOUR: DATA COLLECTION, ANALYSIS AND RESULTS

4.1 Data collection... 76

4.1.1 Model formulation.. 76

4.1.2 Results.. 79

4.2 Discussion 81

CHAPTER FIVE: CONCLUSION AND RECOMMENDATION

5.1 Conclusion.. 83

5.2 Recommendation.. 83

REFERENCES.. 84

APPENDIX A 88

APPENDIX B 92

APPENDIX C 96

APPENDIX D 99

APPENDIX E 101

LIST OF TABLES

Table *Page Number*

Table 1.1: Bus types and their capacities.. 4

Table 3.1: Data for the production and inventory control problem............... .. 33

Table 3.2: Summary of results for March.. 40

Table 3.3: Summary of results for February.. ... 42

Table 3.4: Summary of results for January.. ... 43

Table 3.5: Summary of results for the optimal.. ... 44

Table 3.6: The road system and costs for the stagecoach problem............... ... 46

Table 3.7: Job data for the manufacturing operation..................................... ... 54

Table 3.8: Summary of the optimal solution of the knapsack problem......... ... 61

Table 3.9: Solution to Stage (4)... ... 66

Table 3.10: Solution to stages (3).. ... 67

Table 3.11: Solution to stages (2).. ... 68

Table 3.12: Solution to stages (1).. ... 69

Table 3.6.1: Data of the illustrative example, stage 1................................ ... 71

Table 3.6.2: Data of the illustrative example, stage 2............................... ... 72

Table 3.6.3: Data of the illustrative example, stage 3.............................. ... 72

Table 3.6.4: Data of the illustrative example, stage 4.............................. 73

Table 3.6.5: State variables for 2 years old equipment.............................. ... 73

Table 3.6.6: Solution of stages (4).. ... 74

Table 3.6.7: Solution of stages (3).. ... 74

Table 3.6.8: Solution of stages (2).. ... 75

Table 3.6.9: Solution of stage (1).. ... 75

Table 3.6.10: The optimal replacement policy and its total cost.................. ... 76

Table 4.1: State variables for MMT bus.. ... 78

Table 4.2: Buses optimal decision variable sequence................................. ... 79

Table 4.3: Replace and Keep action profits/loss at given action years........... 80

LIST OF FIGURES

Figure *Page Number*

Figure 3.1: Schematic representation of the production and inventory control problem as a three – stage dynamic programming problem 34

Figure 3.2: The road system and costs for the stagecoach problem.............. . 45

Figure 3.3: Schematic representation of the stagecoach problem as a five – stage dynamic programming problem....................................... . 47

Figure 3.4: A representation of the optimal solution of the stagecoach problem... .. 53

Figure 3.5: schematic representation of the knapsack problem as a four – stage dynamic programming problem... .. 55

Figure 4.1: Replace and keep profit/loss comparison....................................... 81

DEDICATION

To the entire family

ACKNOWLEDGEMENT

To Almighty Allah belongs the glory, thanks and praises.

I would like to express my profound gratitude to my supervisor, Mr. Kwaku Fokuo Darkwah, for his guidance and support throughout my graduate studies in KNUST. He encouraged me to pursue my goal and work hard to achieve excellence. I particularly appreciate his time and efforts at helping his students and giving them those thought provoking advice whenever the need arose. He has played significant role in both my academic and personal development in KNUST, and his vision, energy and desire for excellent goal achievement has influenced me with lifetime benefits.

To my wife, Serwaah Agyekum and children I appreciate every bit of your understanding, time and support all this while. Special thanks to Almighty Allah for such a blessed family.

I am particularly grateful to Alhaji Abdul Razak Tahiru and Mr. Mohammed Sampson Bala, my uncle and senior brother respectively for their encouragement and support so far. It is without any shadow of doubt that their help has seen me through a much more joyful and colourful life in the past years.

To my mates, I say all the best in your endeavours especially Debrah, Adobah, Antwi-Boasiko and Ajao Grace for your time and sacrifices.

I would also like to thank my mother and mother in-law as well as brothers and sisters for their constant support and countless sacrifices. Without them, I could never accomplish so much and reach this milestone in my life. I dedicate this thesis to them.

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND TO THE STUDY

The timely replacement of buses in the fleet is one of the fundamental programs that serve as a backbone of a successful transport system. Buses are a transit system's most valuable asset because good customer service is dependent on the condition of the fleet. The total cost of the fleet is usually the most expensive asset, even more so than the facilities that house the operation. An aging fleet presents a poor image to the system's customers and the general public. Bus maintenance expenses usually increase as the age of a bus advances thereby triggering replacement.

All transport service providers in Ghana maintain large fleets of equipment. This equipment represents a substantial investment and is a vital set of resources that is used to maintain roads and highways. Managing such a large amount of equipment is an important and difficult challenge of deciding when to replace existing equipment. Such decisions have a clearly documented economic impact, and they also affect the ability of the fleet to provide required equipment when needed.

In particular, the Metro-Mass Transit (MMT) Limited Fleet Services Section provides management of MMT's fleet, which consists of over 1000 pieces of active equipment worth approximately GH₵540 - GH₵590 million. This equipment includes a variety of buses.

1.2 PUBLIC TRANSPORT IN GHANA

Transport in Ghana is accomplished by road, rail, air and water. Ghana's transportation and communications networks are centered in the southern regions, especially the areas in which gold, cocoa, and timber are produced. The northern and central areas are connected through a major road system with some areas relatively isolated.

Road transport is by far the dominant carrier of freight and passengers in Ghana's land transport system. It carries over 95% of all passenger and freight traffic and reaches most communities, including the rural poor and is classified under three categories of trunk roads, urban roads, and feeder roads. The four major means of public transport in Ghana are taxi (cab), "trotro", commuter buses and the train.

It is said that Ghana is the country with most taxis in the world. A taxi will be shared by 4 or 5 passengers and its presence is felt even in the remotes village in Ghana. Taxis are largely saloon cars being run commercially.

The name trotro means coin - coin and refers to the little amount paid to travel from one place to another. Without doubt, the trotro is the most common way to travel for Ghanaians. Mini-buses and Urvans dominate this category of commercial cars.

Commuter buses are used by a few transport organizations to perform long distance transport. As well private companies use commuter buses to collect and deliver staff from and to their homes. MMT operates commuter buses in the main corridors of the cities and to the outskirts. In a nutshell, this is what is called intra-city mass transport.

Railway transport in Ghana is still in a development stage, the southern sector is the only portion of the country benefiting at the moment. The northern part of the country is soon to be connected with the national rail line after the completion of the tracks connecting the south to the north.

1.2.1 BACKGROUND TO METRO MASS TRANSIT LIMITED

The Metro Mass Transit Limited (M.M.T.) is a local bus company which is identified to explore the application of dynamic programming for its bus replacement needs. Established in October 2003, the MMT, owned by both the government of Ghana and private investors is poised to providing an efficient urban mass transport system in Ghana through the use of buses. The company aims at operating an effective and affordable transport system in an economical sustainable way in Ghana and that is characterized by the three bus service systems below:

- **Bus Rapid Transit System** - designed only for the congested roads in Ghana; these are presently the main corridors of Accra and Kumasi.
- **Urban Service** - operates in any greater urban area connecting central bus terminals with city outskirts and provides upon that medium-distance transportation to villages in the surrounding of a regional capital.
- **Rural Bus services** - This long distance rural bus service of MMT operate mainly on rough roads. Because of the long journey, the rural service has a low but solid frequency.

The company's bus terminal in Kumasi is made up of four DAF buses, forty-six VDL Neoplan City buses, thirty VDL Commuter buses and fourteen VDL Jonckheere buses that operate on six intercity routes; twelve inter urban / rural urban routes and thirteen intra city routes. Table 1.1 summarizes the fleet size of each bus type with their corresponding capacity respectively.

Table 1.1: Bus types and their capacities

BUS TYPES	NO. OF BUSES	BUS CAPACITY
DAF	4	63
VDL Neoplan City	17	37
(1st generation)		
(2nd generation)	29	47
VDL Commuter	30	63
VDL Jonckheere	14	62

1.3 DESCRIPTION OF BUS MAINTENANCE

Regular Preventive Maintenance encompasses that each bus has a regular oil changes as specified by the manufacturer. Buses also maintain annual state inspection. The regular maintenance contributes to the efficiency of bus serviceability. Normal preventive tasks include the following: state inspection, as required by the law; oil changes, as stated by the manufacturer of the bus; tune-up, as stated by the manufacturer of the bus; and minor maintenance and safety items like wiper, bulbs, etc., as needed.

Oil changes and minor repairs are carried out in a timely fashion at the specified bus maintenance facility. MMT has selected a bus repair sub-contractor close to the research facility for these tasks. Estimated time for this service is one and a half hours including all travel times.

Major maintenance on any bus failure not covered under regular preventive maintenance is defined as a major failure event. Currently there is no established assessment policy for major maintenance. Estimated repair time for major maintenance work is on an average 8 hours. During bus downtime repairmen are highly constrained in carrying out their tasks. A bus needing major maintenance is repaired as needed. The company is in outsourcing partnership agreement with Neoplan Ghana Ltd to take care such situations.

Catastrophic failure on any bus is when the bus is out of commission given that the estimated repair cost is high and possibly exceeds the future benefits from the usage of the bus in question. There is no formal system in place for estimating the future value of the bus. However, if it is felt by the bus supervisor that cost of repair is too high, it is considered catastrophic failure and such an event triggers an automatic bus replacement process.

1.4 PROBLEM STATEMENT

Prior to this study, MMT Ltd was challenged with how buses could be rated for replacement purposes but to a large extent based any such decision on the buses expected useful life (economic life span). These decisions were meant to ensure that buses purchased with MMT funds are maintained and remained in transit use for a minimum normal service life.The scenario has a lot of validity considerations both within Fleet Services and with the various bus crews that receive and evaluate its output.

Data from MMT Fleet Services clearly showed that for many large equipment classes, newer equipment was being utilized more than older equipment. As an example of how this may occur, it may be common for users of passenger buses in a fleet to request newer buses to hire when they are available. This decreasing utilization of older equipment was occurring as the overall service provided by the fleet stayed constant. The end effect of this was that replacement decisions not only affected the specific equipment being replaced, but also the utilization of other equipment in the same class (assuming the replacement is new). Furthermore, it was known that reduced utilization of a single piece of equipment as it ages extended the equipment's economic life. This research examined how these facts affected MMT Ltd's net profit.

1.5 OBJECTIVE OF THE STUDY

The objectives of this work are:

1. To model operational costs of MMT Ltd as a recursive function.
2. To solve for an optimal policy for replacing MMT Ltd's buses using dynamic programming technique.

1.6 METHODOLOGY

MMT is faced with a replacement problem and a dynamic programming method which usually refers to simplifying a decision by breaking it down into a sequence of decision steps over time was used. The backward cost/profit recursive algorithm which solves automobile replacement problems of DP kind was employed and implemented using Microsoft Excel solver.

Data on the types of buses, replacement cost of a buses, maintenance cost of buses and income generated (yr) by each bus was obtained from the workshop manager and the statistical office of the company respectively.

The internet to a large extent used in obtaining the relevant and related literature. Books from the main Library at KNUST and the Mathematics Department's library were thoroughly read in the course of the project.

1.7 JUSTIFICATION

The accomplishment of the dynamic programming based automobile replacement policy stated will assist MMT and other Transport Service Providers nationwide to better access and manage equipment needs particularly replacement. The creation of a more effective equipment replacement system will be of tremendous benefit both in potential labour and equipment cedi savings. Additionally, it will be possible to identify the limitations of current research when considering the real-world characteristics and availability of data.

1.8 THESIS ORGANIZATION

Chapter one covers the background to the study and public transport in Ghana, brief discussing of the methodology and the objective of the study were also handled in this chapter. The Literature review is contained in chapter two with chapter three solely devoted to the methodology/research approach adopted for this study.

Chapter four contains data collection, analysis and discussion whereas the conclusions and recommendations are dealt with in chapter five**.**

CHAPTER TWO

LITERATURE REVIEW

The purpose of this study was to conduct a dynamic programming survey and come to terms with the state-of-the-art in equipment replacement models in published research literature as well as in practice.

2.1 DYNAMIC PROGRAMMING REVIEW

The term dynamic programming was originally used in the 1940s by Richard Ernest Bellman to describe the process of solving problems where one needs to find the best decisions one after another (Adda et al, 2003).

Slater (1964) uses dynamic programming to determine an optimal path from a number of alternatives paths, in order to move from a given initial state to a desired final position.

In identifying an optimal strategy for finding a solution to a contract bridge tournament, Beaumont (2007) used dynamic programming to accomplish this task. The contract bridge tournament comprises several rounds of matches in which players compete as pairs for 'master points' awarded for each match won or drawn and for being highly placed at the end of the tournament. In the second and subsequent rounds, pairs are matched against other pairs that have been approximately equally successful so far. The optimal strategy is a function of a pair's ability.

The best-scoring set of beat times that reflects the tempo as well as corresponding to moments of a high 'onset strength' in a function derived from audio was found using dynamic programming as seen in Daniel (2007). This very simple and computationally efficient procedure is shown to perform well on the MIREX-06 beat tracking training data, achieving an average beat accuracy of just fewer than 60% on the development data.

Nicole and Quenez (1995) also used to determine a solution for the problem of pricing contingent claims or options from the price financial market. In this situation, there is a price range for the actual market price of the contingent claim. The maximum and minimum prices are studied using stochastic control methods. The main result of this work is the determination that the maximum price is the smallest price that allows the seller to hedge completely by a controlled portfolio of the basic securities. A similar result is obtained for the minimum price (which corresponds to the purchased price).

Bush, et al (1990) describes a compile-time analyzer that detects dynamic errors in large, real – world programs. The analyzer traces execution paths through the source code, modeling memory and reporting inconsistencies.

Zeqing and Shin (2006) introduced and studied properties of solutions for functional equations arising in dynamic programming of multistage decision processes.

Quansong, et al (2006) in their studies identified the microbial community composition and its variations in environmental ecology using dynamic programming. Clustering analysis of the Automated Ribosomal Interagency Spacer Analysis (ARISA) from different times based on the dynamic programming algorithm binned data revealed important features of the biodiversity of the microbial communities.

Stochastic dynamic programming model was used by Norman and Clarke (2004) to examine the appropriateness of sending a lower order batsman into 'hold the fort' on a 'sticky wickets'. In cricket, a rain-affected pitch can make batting more difficult than normal.

Several other conditions such as poor light or an initially lively pitch may also result in difficulties for the batsman. All these are referred to us ‘sticky wickets’.

Dynamic programming was used to get an optimal price for a car of a professor who had limited number of days to leave a country after his sabbatical leave. Mahmut (2000) details this classical dynamic programming application.

2.2 REPLACEMENT PROBLEMS REVIEW

Fleet managers and researchers in their bid to addressing the problem of equipment replacement identified long ago, developed a variety of strategies. In order to complete a comprehensive and a thorough overview of developed approaches, published models and studies were reviewed and a survey was carried out to answer how replacement problems are managed in practice at various Transport service providers. This approach revealed among other things a difference between theory and practice.

This assessment focused on equipment replacement studies and research that are applicable or motivated by replacement for bus fleets. The main question that was addressed was how to identify replacement candidates among fleet members so that total fleet costs are minimized in the long run. It is worth noting however that equipment replacement dates back from two early works of (Taylor, 1923; Hotelling, 1925). Taylor in his paper developed by means of a discrete period analysis, a formula relating the average unit cost of the output of a machine over L years (the years of machine life) to the cost of a new machine, the scrap value of the machine after L periods of service, the operating costs of the machine in each period of service up to the L period, the output of the machine in each period, and the rate of interest.

The manufacturer’s desire to make his unit cost a minimum or that consideration of profit led him to scrap the machine at some different point in time from that which makes the unit cost a minimum remained the key challenge that propelled Hotelling's different dimension to Taylor's preposition. He advances the view point that the owner of the machine wishes to maximize the present value of machine's output minus its operating costs.

Preinreich (1940) explained that the economic life of a single machine could not be determined in isolation from the economic life of other machines in the chain of future replacements extending as far as into the future as the firm's profit horizon. He argued that the firm should maximize the present value of the 'aggregate goodwill' of all replacement, where the goodwill is the present value of earnings of the future machine, replacements minus the present value of costs of all such machines.

An intuitive method for identifying replacement candidates is to define a replacement standard such as an equipment age standard. Assets that exceed the age standard are candidates for replacement. A ranking can then be implemented that sorts equipment units by how much they exceed the standard. One of the most popular approaches to derive an age standard is the application of single asset replacement analysis to compute an "economic life," which is also known as life cycle cost analysis (LCCA). LCCA is extensively covered in the engineering economics literature. Eilon et al. (*1966*) considered acquisition cost, resale value and maintenance cost in order to derive the minimum average costs per equipment year and the corresponding optimal equipment age policy for a fleet of fork lift trucks. Chee (*1975*) analyzed the fleet of Ontario Hydro using LCCA and generated optimal equipment age policies for different equipment classes. Chee proposed to also consider repair costs for individual equipment units given that LCCA gives only one replacement criterion– namely the economic life – for a whole equipment class. As a result, repair cost limits are computed in addition to an economic life. If a fleet member stays within the repair cost limits for each year, it is replaced only after reaching the economic life of its class.

Weismann et al. (*2003*) applied LCCA to individual pieces of equipment in the Texas DOT fleet. Their results indicated that this approach combined with a multi-attribute ranking is more cost efficient than utilizing a single age standard. This multi-attribute ranking considers economic life, operation costs, repair costs and usage in order to assign replacement priorities to equipment units.

Ayres and Waizeneker (*1978*) normalized annual maintenance costs by mileage and current acquisition costs, and then used this inflation-independent parameter for LCCA. The normalization is assumed to fix the problem of differences in complexity and function of equipment units. Thus, the method can make replacement decisions fleet-wide – ignoring the fact that a fleet consists of different equipment classes.

Another popular replacement criterion utilized was repair costs. Some literature provides evidence that repair cost limit policies have some advantages over lifetime limit policies.

Data for army buses was analyzed by Drinkwater and Hastings (*1967*) where they derived age dependent frequencies for repair visits per year and distributions for repair costs per visit. To determine optimal repair cost limits, they used this information in a combination of dynamic programming and Monte Carlo simulation and it shown that their repair cost limit policy leads to financial savings when compared to an LCCA-based economic age policy, and also when compared to an experience-based repair cost limit policy (which was previously applied on the army fleet). Love et al. (*1982*) came out with similar results having worked with fleet data from Postal Canada and compared economic age policies with repair cost limit policies. They derived economic ages analytically and repair cost limits were generated in a Markov simulation. Applied to the Postal Canada fleet, the repair cost limit policy was superior to the economic age policy.

Instead of using repair cost limits for repairs that have occurred, Hastings (*1969*) derived repair cost limits for estimates of future repair costs. He assumed that before any repair measure was conducted, fleet members were run through an inspection and repair costs were estimated. The actual repair was only undertaken if estimated costs were smaller than the derived repair cost limit.

Nakagawa and Osaki (*1974*) in a much more different approach did not focus on repair costs, but on repair time. Their policy was characterized by defining a limit for the time a broken unit of equipment spends in repair measures. Minimizing expected costs per unit time over an infinite time span yielded the repair time limit as per its derivation.

The problem of optimal replacement to the problem of optimal buy, operate and sell policies has been expanded by other approaches. Simms et al. (*1984*) detailed data from an urban transit bus fleet. Equipment units in this fleet were operated at different levels and performed different tasks as a function of age or cumulative mileage, subject to varying capacity constraints. Consequently, newer equipment units had different acquisition and operating cost structures than older less sophisticated fleet members. By applying a combination of dynamic programming and linear optimization, an optimal buy, operate and sell policy was derived for the investigated fleet.

Hartman (*1999*) in a similar fashion as Simms et al looked for the minimum cost replacement schedule and associated utilization levels for a multi-asset case – emphasizing that utilization is a decision variable and not a parameter. The author examined the problem of simultaneous determination of asset utilization levels as well as replacement schedules, while the total costs of assets that operated in parallel were minimized. A linear program that considered dependency of operating costs on utilization levels and dependency of utilization levels on a deterministic demand solved the problem.

In later works, Hartman was encountered with the same challenge, but asset utilization levels had to meet a stochastic demand *(Hartman 2004)*. With two equipment units and parallel operation of both assets in a much more simplified case, the author determined the optimal replacement schedules and utilization levels for both individual buses by applying dynamic programming. Both Simms and Hartman faced complex equipment replacement, operating and scheduling problems in bus fleets. They did not promote particular replacement criteria but presented optimization methodologies that led to cost efficient results for a specific fleet.

Previous works reviewed specifically did not consider decreasing utilization levels of assets as they age. At MMT, equipment utilization has been decreasing with equipment age, but constant utilization has been a widely spread assumption made in the replacement models literature.

Simms et al. (*1984*) derived an optimal buy, operate and sell policy for an urban transit bus fleet whose members operated at different levels depending on equipment age. They reduced the problem to two levels of utilization: young buses were operated at a constantly high level meeting the base demand, while utilization was constantly low for buses older than ten years because they were only used when needed to meet peak demand. Unlike the replacement decision at other transport service providers however, they assumed utilization was controllable.

Redmer (*2005*) derived the optimal lifetime limit or economic life for freight transportation fleet, which showed decreasing utilization as equipment grew older and constant utilization levels within age classes. The basis of his model was the LCCA approach from Eilon et al. (*1966*), which assumed constant utilization, and thus, was not directly applicable to the fleet considered. Eilon et al. considered analyzed costs per unit time. Redmer concluded that Eilon’s model provided lifetime limits approaching infinity when the fleet data showed decreasing utilization with age. Instead of using costs per unit time, Redmer modified Eilon's LCCA approach so that costs were given per kilometer. As a result, discounted costs of ownership per kilometer were minimized over replacement age and a feasible, cost minimizing economic life was provided.

The second study underlining the importance of decreasing utilization levels over equipment age was published by Buddhakulsomsiri and Parthanadee (*2006*). Their model was adopted from Hartman (*1999*). A major difference was that in Hartman's model, utilization was defined as a decision variable, whereas in Buddhakulsomsiri and Parthanadee's study it was assumed that utilization per age class was constant, and thus utilization was a model parameter. Their assumptions about utilization levels were identical to the assumptions made by Redmer. In addition, Buddhakulsomsiri and Parthanadee explained that decreasing utilization might follow from a dependent use pattern: "Given that the various buses are available to provide the same service or perform the same function, it is the newer ones that are generally preferred."

Eventually, by minimizing the total costs of purchasing, selling, owning, and operating equipment units over a finite planning horizon Buddhakulsomsiri and Parthanadee provided a fleet specific and cost minimal buy, operate, and sell policy.

Problems related to equipment replacement in fleets were analyzed by Khasnabis et al. (*2003*), Davenport et al. (*2005*) and Rees et al. (*1982*). Rees et al. made a replacement demand forecast by simulating the steady process of deterioration and equipment breakdown within a Markov type network. Davenport et al. on their part created a fleet condition forecast model for a fleet of cutaway passenger vans by using a regression model they found out that, the parameters equipment age, total mileage, miles per year on unpaved roads, lift equipment, and percentage of population older than age 65 were the best equipment condition predictors.

With the assumption that future demand for fleet services and the expected costs of replacement, rehabilitation and remanufacturing were known, Khasnabis et al. showed that the optimal capital allocation for the dual purpose of purchasing new equipment units and rebuilding existing ones within the constraint of a fixed budget could be obtained with linear programming.

The available literature on discrete time maintenance models predominantly treats an equipment deterioration process as a Markov chain. Sherwin and Al-Najjar (1999) presented a Markov model to determine the inspection intervals for a phased deterioration monitored complex components in a system with severe down time costs. An example involved roller bearing in paper mills with three phases; no defect, possible defect and final deterioration towards failure. In the last phase, continuous monitoring was used. The output of the model was an optimum inspection rate for each phase given a switching rule for going over to continuous monitoring. Wang and Hwang (2004) presented a Markov model that could be applied to construct the relationships among maintenance cycle, maintenance personnel allocation, human recovery factor, and system's tolerance time. Zhou et al. (2006) presented a dynamic opportunistic condition-based predictive maintenance policy for a continuously monitored multi-unit series system that was proposed based on short-term optimization with the integration of imperfect effect into maintenance actions. In their research, it was assumed that a unit's hazard rate distribution in the current maintenance cycle could be directly derived through CBPM. Whenever one of the units fails or reaches its reliability threshold, the whole system has to stop and PM opportunities arise for the system units. Jardine et al. (1997) presented an optimal replacement policy based on Markov stochastic process. Gupta and Lawsirirat (2006) presented a simulation based optimization method for strategically optimum maintenance of monitoring-enabled multi-component systems using continuous-time jump deterioration models.

Sherwin (1999) with the concept of opportunity maintenance suggests new ways to construct and update preventive schedules for a complex system by making better use of system failure down time to do preventive work. Sinuany-Stern et al. (1997) concentrated on the 2-action version of this preventive schedules problem. They suggested an extremely practicable decision rule in partial observability, and proved empirically that this rule more than satisfactory competes with the state of- the-art generic algorithm when implemented with its recommended grid usage. Sinuany-Stern (1993) considered a production system (machine) which deteriorates over time and the system deterioration over time was assumed to be Markovian. Moreover, the time scale assumed discrete and the 'true' state of the system (excellent, medium and bad) was not directly observable. What is observed was the performance of the system measured in terms of 'number of defectives' per time period. At the end of each period, a decision was to be made: whether to replace the system or not and the objective was to minimize the total cost in the long run.

CHAPTER THREE

METHODOLOGY/ RESEARCH APPROACH

This chapter looks at optimization techniques, dynamic programming technique and its application to various problems, survey of replacement models.

3.1 OPTIMIZATION TECHNIQUES

Optimization techniques are designed to maximize profit or minimize cost of any business operation. There are specialized techniques under the optimization model for specific problems.

The models include:

i. Linear Programming: It is best handled by the simplex algorithm, and also solves linear models. Linear programming (LP) is a technique for optimization of a linear objective function, subject to linear equality and inequality constraints. Linear Programming determines the way to achieve the best outcome (such as maximum profit or minimum cost) in a given mathematical model, given some list of requirements represented as linear equations (Alexander, 1998).

ii. Integer Programming: It solves the same mathematical model as that of linear programming, but with the additional restriction that some of the decision variable must have integer values.

iii. Dynamic programming: Dynamic programming works on the principle of finding an overall solution by operating on an intermediate point that lies between where we are now and where we want to go. Since the intermediate point is a function of the point already visited, the procedure is said to be recursive.

3.2 RECURSIVE PROGRAMMING

Dynamic programming and many useful algorithms are recursive in structure. In solving a given problem the algorithm calls a subroutine recursively one or more times to deal with closely related sub-problems. These algorithms typically follow a divide-and-conquer approach in the sense that they break the problem into several sub-problems that are similar to the original problem but smaller in size. The sub-problems that are similar to the original problem but smaller in size are solved recursively, and then these solutions are combined to create a solution to the original problem.

3.2.1 DIVIDE-AND-CONQUER ALGORITHM

The divide-and-conquer paradigm is a recursive algorithm and it involves three steps at each level of the recursion.

i. Divide the problem into number of sub-problems.

ii. Conquer the sub-problems by solving them recursively. If the sub-problem sizes are small enough, however, just solve the sub-problems in a straightforward manner.

iii. Combine the solutions to the sub-problems into the solution for the original problem for example, consider the minimization problem below:

Minimize $f(x) = X^4 - 5X + 2X$ subject to $-1 \leq X \leq 1$, by reducing the interval of uncertainty to less than 10% of the original and using the Fibonacci search algorithm with

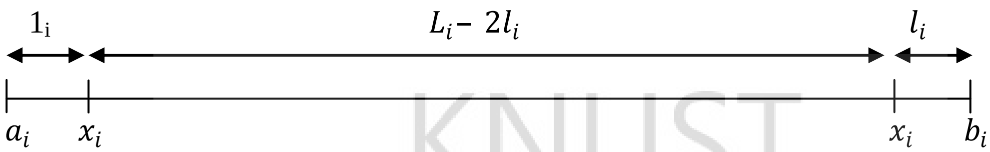
$F_0 = F_1 = 1$; $F_{n-2} + F_{n-1} = F_n, n \geq 0$ (Amponsah, 2006).

Thus $\quad F = [1, 1, 2, 3, 5, 8, 13, 21, \ldots]$

We choose n such that 1< 10 =0.1. Now for n =6, 1 = 1 = 0. 076923 < 0.1 and hence

We shall make six applications of the Fibonacci numbers as follows:

Let [a, b] = [-1, 1], then $L_1 = b_1 - a_1 = 1 + 1 = 2$. Using the formula $l_i = \frac{F_{n-(i+1)}}{F_{n-(i-1)}} Li$ calculate the interval of reduction 1i such that the point x_i and y_i divide Li into three sections with $x_i = a_i + l_i$ and $y_i = b_i - l_i$, $i = 1,2,\ldots$



Evaluate $f(x_i)$ and $f(y_i)$ and select the point that gives the minimum evaluation.

For sub problem $i = 1, n = 6,\ L_1 = 2$.

Using the formula $l_i = \frac{F_{n-2}}{F_n} Li,\quad l_1 = \frac{5}{13} \times 2 = 0.76923.$

$Hence\ x_1 = a_1 + l_1 = -1 + 0.76923 = -0.23077\ and\ y_1 = b_1 - l_1 = 1 - 0.76923$

$= 0.23077.$

$f(x_1) = 3.15668, f(y_1) = 0.848985$. Since $f(x_1)$ we discard $[a1, x1]$ and set

$a_2 = x_1, \quad b_2 = b_1$

For sub problem $i = 2$

Put $[a_2, b_2] = [-0.23077, 1]$ and $L_2 = 1 + 0.23077 = 1.23077$

$$l_2 = \frac{f_{6-3}}{f_{6-1}} L_2 = \frac{3}{8} \times 1.23077 = 0.461538.$$

Hence $x_2 = a_2 + l_2$= -0.23077+0.461538 = 0.23007 and y_2= $b_2 - l_2$ = 10.461538=

0.538462, $f(x_2) = 3.15668$ and $f(y_2) = -0.608245$.

Since $f(x_2) > f(y_2)$ we discard the interval $[a_2, x_2]$ and put $[x_3, b_3] = [0.23077,1]$.

For sub problem $i = 3$

$$L_3 = b_3 - a_3 = 0.76923 \; and \; l_3 = \frac{F_{6-(3+1)}}{F_{6-(3-1)}} L_3 = \frac{2}{5} \times 0.76923 = 0.307692.$$

Hence $x_3 = a_3 + I_3 = 0.229719 + 0.3076923 = 0.538462 = y_2$

and $y_3 = b_3 - l_3 = 1 - 0307692 = 0.692308.$

$f(x_3) = -0.608245, f(y_3) = -1.23182, since\ f(x_3) > f(y_3)$ we discard the interval $[a_3, x_3]$ and $[a_4, x_4]$ =[0.58462, 1].

Continuing we have the as final interval of uncertainty to be and min. $f(x_6) = -1.71814$ occurs as $x_6 = 0.846154$ (Amponsah, 2006).

The interval of uncertainty currently becomes the search domain for the solution of the current sub problem. After the solution of each sub problem the interval of uncertainty is reduced further.

The interval of uncertainty is a division of the original domain and that solution in that interval is the conquest of the interval. This continues until we obtain the final interval of uncertainty that satisfies the formulation condition. The optimal solution to the original problem is then determined. Thus the optimal solution is a conquest of the final part division of the original domain by use of the recursive formula.

3.2.2 GREEDY ALGORITHM

A greedy algorithm is a recursive algorithm that follows the problem solving heuristic approach and makes locally optimal choice at each stage in the computation with the hope of finding the global optimum.

An optimization problem can be solved by greedy algorithm, if the problem has two ingredients (properties):

(i) Greedy choice property

A globally optimal solution can be arrived at by making a locally optimal (greedy) choice thereby remains the most vital ingredient. In order words when we are considering which choice to make, we make the choice that looks best in the current stage of the problem, without considering the results from subsequent choices to be made. Here is where greedy algorithms differ from dynamic programming. In dynamic programming, we make a choice at each step, but the current choice usually depends on the solutions to previous sub problems. Consequently, we typically solve dynamic programming problems in a bottom-up manner, progressing from smaller sub problems to larger sub problems. In a greedy algorithm, we make whatever choice seems best now and then solve the sub problem arising after the choice has been made. The choice made by a greedy algorithm may depend on choices so far, but cannot depend on any future choices or on the solutions to other sub problems. Thus, unlike dynamic programming, which solves the sub problems bottom up, greedy strategy usually progresses in a top-down fashion, making one greedy choice after another, reducing each give problem instance to a smaller one.

ii. Optimal Substructure

A problem exhibits optimal substructure if an optimal solution to t he problem contains within it optimal solutions to sub problems. This property is a key ingredient of assessing the applicability of dynamic programming as well as greedy algorithm. Optimal substructure varies across problem domains in two ways.

a. The number of sub problems used in the process of computing the optimal solution to the original problem, and

b. The number of choices available in determining which sub problem(s) to use in an optimal solution process.

AN ILLUSTRATIVE EXAMPLE

A customer went to the sorcery shop. He paid for the items bought and was to receive a change of 41 cents. However the sales clerk had the following denomination of coins:

i. 25 cents (quarter) denominations

ii. 10 cents (dime) denominations.

iii. 5 cents (nickel) denominations

iv. 1 cent denominations.

The sales clerk is to give the minimum number of coins that will be equal the change of 41 cents.

The problem is which denominations should she select and how many coins of each selected denomination should be used to give the minimum of coins for the 41 cents change.

Greedy choice option: The coin with the highest denomination is chosen at each step.

Optimal sub structure: The problem of selection of a coin denomination is a sub problem. The choice of highest coin denomination possible is an optimal solution to the sub problem. Since the solution of the original problem is the count of the number of coin denominations selected at the various sub problems the problem possesses an optimal substructure.

Sub problem 1:

Select coin to reduce change of 41 cents.

Greedy solution: Choose highest coins denomination of 25 cents

Solution of sub problem 1 is 1 coin of 25 cents.

The remaining change is 41-25=16

Sub problem 2:

Select coin to reduce 16 cents.

Greedy solution: Choose highest coins denomination of 10 cents.

Solution of sub problem 2 is 1 coin of 10 cents.

The remaining change is 16-10=6

Sub problem 3:

Select coin to reduce change of 6 cents.

Greedy solution: Choose highest coins denomination of 5 cents

Solution of sub problem 3 is 1 coin of 5 cents.

Remaining change: 6-5=1.

Sub problem 4:

Select coin to reduce change of 1 cent.

Greedy solution: Choose highest coins denomination of 1 cent solution of sub problem 1 is 1 coin of 1 cent.

Remaining change: 1-1=0

The sales clerk should give one 25 cents, one 10 cents, one 5 cents and one 1 cent as the change becomes the solution.

3.3 PRINCIPLES OF DYNAMIC PROGRAMMING

Dynamic programming was the brainchild of an American Mathematician, Richard Bellman, who described the way of solving problems where you need to find the best decisions one after another. The word *Programming* as in Bellman (1957) and Bhowmik (2010) indicate that the name has nothing to do with writing any code or computer programs. Mathematicians use this speech to illustrate a set of rules which anyone can follow to solve a problem. 'They do not have to be written even in a computer programming language' (David and Pass, 1997; Coremen et al., 2008). The word "programming" in "dynamic programming" is a synonym for optimization and is meant as "planning or a tabular method". It is basically a stage wise search method of optimization problems whose solutions may be viewed as the result of a sequence of decisions as elaborated in Bhowmik (2010).

General working methodology for achieving solution using this approach is given as:

i. *Divide into Sub problems*

The main problem is divided into a number of smaller, similar sub problems. Alsuwaiyel (2002) and Bhowmik (2010) maintain that the solution to main problem is expressed in terms of the solution for the smaller sub problems. Stage wise solutions start with the smallest sub problems. ii. *Construction of Table for Storage*

The underlying idea of dynamic programming is to avoid calculating the same stuff twice and usually a table of known results of sub problems is constructed for the purpose.

Bhowmik (2010) and Howard (1960) stress that dynamic programming takes advantage of the duplication and arrange to solve each sub problem only once, saving the solution in table for later use. The key to competence of a dynamic programming algorithm is that once it computes the solution to a constrained version of the problem, it stores that solution in a table until the solution is no longer needed by any future computation. The initial solution is trivial as in Vijaya (2006). This tells us that we trade space for time to avoid repeating the computation of a sub problem.

iii. Combining using Bottom-up means

Combining solutions of smallest sub problems obtain the solutions to sub problems of increasing size. Horowitz et al. (2008) and Bhowmik (2010) reiterate that the process is continued until we arrive at the solution of the original problem.

Bhowmik (2010) and Tsitsiklis and Roy (1999) look at dynamic programming involving selection of optimal decision rules that optimizes a certain performance criterion:

i. *The Principle of Optimality* – An optimal sequence of decisions is obtained iff each subsequence must be optimal. That means if the initial state and decisions are optimal then the remaining decisions must constitute an optimal sequence w.r.t the state resulting from the first decision. According to Bellman (1957), combinatorial problems may have this property but may exploit too much memory and/or time towards efficiency.

ii. *Polynomial Break up* - The original problem is divided into several sub problems. The division is done in such a way that the total number of sub problems to be solved should be a polynomial or almost a polynomial number. This is done for efficient performance of dynamic programming.

Using the top-down view of dynamic programming, the first property mentioned above corresponds to be able to write down a recursive procedure for the problem that we want to solve. The second property makes clear in our mind that this recursive procedure builds only a polynomial number of different recursive calls as clearly seen in Ross (1983).

3.3.1 DYNAMIC PROGRAMMING APPLICATIONS

The versatility of the dynamic programming method is really appreciated by exposure to a wide variety of applications. My intent is to understand and contribute to the equipment replacement research on optimization problems. We provide examples in subsequent sections to illustrate some of the varied problems that dynamic programming can solve. These are;

i) Production and inventory control problem,

ii) The stagecoach problem(network problem),

iii) Knapsack problem and

iv) The equipment replacement problem.

3.3.2 DYNAMIC PROGRAMMING STRENGTHS

Creativity is necessary before we can distinguish that a particular problem can be casted effectively as a dynamic program. Kleinberg (1962) and Howard (1960) consider an even clever insights to restructure the formulation often are essential in useful solution. This idea of reusing sub problems is the main advantage of the dynamic programming paradigm over recursion. The simplicity that makes dynamic programming more appealing is both a full problem solving method and a subroutine solver in more complicated algorithmic solutions, Weimann (2009) and Streufert (1998) throw more light on this. The key to competence of the dynamic programming approach lies in a table that stores partial solutions for future references. Attractiveness of dynamic programming during the search for a solution on the other hand lays avoidance of full enumeration by clipping early partial decision solutions that cannot possibly lead to optimal solution Nature, Blackwell (1965), Bergin (1998) and Streufert (1998) make it clear in a single word that makes the optimization procedure multistage in.

The most charisma involves selection of optimal decision rules: *The Principle of Optimality* and *Polynomial Break up*, which optimizes performance criterion. The approach is both a full problem solving method and a subroutine solves. This piece is to a large extent evidenced in Bhowmik (2010), Skiena (1999) and Ross, (1983). These simplicities make dynamic programming technique more appealing in complicated algorithmic solutions that also we think about.

Dynamic programming is so powerful device that encourages tremendous growth in researches for solving sequential decision problems, and research related to dynamic programming has lead to fundamental advances in theory, numerical methods, and econometrics. Thus, dynamic programming can be sighted as a useful "first approximation" scheme to human decision making. Rust (2006), Traub and Werschulz (1998) narrate that it will undoubtedly in near future be old-fashioned by more descriptively accurate psychological models.

Finally, Chinneck (2006) has it that though it is tedious to accomplish by hand, but dynamic programming is actually relatively efficient compared to a brute force listing of all possible combinations to find the best one.

3.3.3 DYNAMIC PROGRAMMING SHORTCOMINGS

The kinds of problems solved using Dynamic Programming are without any shadow of doubt optimization problems. But the optimal solution involves solving a sub problem, and then it uses the optimal solution to that sub problem, Ross (1983) explains. This key property of the solutions produced by dynamic programming is that they are *time consistent.* This is essentially due to direct implication of the principle of optimality as clearly indicated in Rust (2006). Another drawback of this practice is that it works best on objects which are linearly ordered and cannot be rearranged such as characters in a string, points around the boundary of a polygon, matrices in a chain, the left-to-right order of leaves in a search tree, Bellman (1957) and Weimann (2009) outline. The major shortcoming of making use of dynamic programming as a means is that it is often nontrivial to write code that evaluates the sub problems in the most efficient order as seen in Wagner (1995) and Howard (1960). The challenge of devising a good solution method is in steps forward to make decisions what are the sub problems, how they would be computed and in what order. Apart from the obvious requirements - *The Principle of Optimality and Polynomial Break up* in Bhowmik (2010) and Weimann (2009) is an efficient dynamic programming which induces only a “small” number of distinct sub problems.

3.3.4 COMPARATIVE ADVANTAGE

DP approach is by far the most powerful optimization paradigm over the others. But its popularity stems from the comparative study with other two popular techniques Divide-and-

Conquer and Greedy Method carried out in Horowitz et al. (2008) and Bhowmik (2010). Like divide-and-conquer, dynamic programming results optimal solutions by combining the partial best possible solutions to sub-problems. Unlike the case in divide-and-conquer algorithms, immediate implementation of the recurrence results in identical recursive calls that are executed more than once, Alsuwaiyel (2002) explains. The structure of dynamic programming is similar to divide-and-conquer, except that the sub problems to be solved are overlapping in nature which makes as a consequence different recursive paths to the same sub problems, Chow and Tsitsiklis (1989) indicates. Thus, for solving a problem, divide-and-conquers is Independent sub-problems, solve sub-problems independently and recursively. Conversely, in dynamic programming sub problems are dependent. Greedy method is also a powerful technique for optimizations but not much like dynamic programming approach. In greedy, we solve a problem making greedy choices. After the choice is made the sub problem is arising. These choices may depend on previous choices. However, the choice is independent of the solutions to sub problems as seen in Coremen (2008) and Vijaya (2006). Top-down convention is normally used towards the feasible solution decreasing current problem size. Unlike greedy, choice is made at each step and bottom up approach is employed increasing problem size from smaller to larger sub problems answering optimal solutions. Bhowmik (2010), Chinneck (2006) and Wilf (1994) clearly indicate that it is more powerful than greedy as it could be applicable to wide range of applications.

3.4 A PRODUCTION AND INVENTORY CONTROL PROBLEM

We consider a minimization problem where we minimize the sum of the production cost and inventory holding cost over a three – month period subject to demand, production capacity, warehouse capacity and inventory holding capacity. At any period, the ending inventory will be calculated as:

$Ending\ inventory\ =\ beginning\ inventory\ +\ production - demand,$ during the period the total cost for each period is the sums of production accost and inventory holding cost for the month and is to be minimized for each period and over the entire duration.

The ending inventory which serves as the first constraint must be less than or equal to the warehouse capacity. The second constraint is that the production level in each period must not exceed the production capacity and the third constraint remains that the beginning inventory plus production must be greater than or equal to demand.

Suppose that we have developed forecasts of the demand for cars over three months and that we would like to decide upon a production quantity for each of the periods so that demand can be satisfied at a minimum cost. There are two costs to be considered: production costs and inventory holding costs. We will assume that production setup costs will be made each period and that setup costs will be constant. As a result costs are not considered in the analysis.

We consider allowing the production and inventory holding costs to depend on quantity at hand and vary across periods. This makes our model more flexible since it also allows for the possibility of using different facilities for production and different storage capacity constraints, which may vary across periods David, et al (1988) explains.

Step 0: Variable definitions and data

Let us adopt the following notation:

$N\ =\ number\ of\ periods\ (stages\ in\ or\ dynamic\ programming\ formulation)$

$Dn\ =\ demand\ during\ stage\ n;\ n\ =\ 1, 2 \dots\ N.$

$X =\ a\ state\ variable\ representing\ the\ amount\ of\ inventory\ on\ hand\ at\ the$

$$beginning\ of\ state\ n;\ n\ =\ 1, 2, \ldots\ N.$$

$$dn\ =\ decision\ variable\ for\ storage\ n. It\ is\ the\ production\ quantity\ for\ the$$

$$corresponding\ \ period\ n:$$

$$Pn\ =\ production\ capacity\ in\ stage\ n:$$

$$Wn\ =\ storage\ capacity\ at\ the\ end\ of\ stage\ n;$$

$$Cn\ =\ production\ cost\ per\ unit\ in\ stage\ n;$$

$$Hn\ =\ holding\ cost\ per\ unit\ of\ ending\ inventory\ for\ state\ n.$$

Table 3.1 labels column one as the month, column two the demand (Dn) for the month, column three the production capacity (Pn), column four the storage capacity (Wn) column five the production cost per unit (Cn) and column six is the holding cost per unit (Hn) for the month.

Table 3.1: Data for the production and inventory control problem

Month	Demand (Dn)	Production capacity (Pn)	Storage capacity (Wn)	Production cost per unit (Cn)	Holding cost per unit(Hn)
January	2	3	2	\$175	\$30
February	3	2	3	\$150	\$30
March	3	3	2	\$200	\$40

The beginning inventory for January is one unit. We will develop the dynamic programming solution for the problem covering N = 3 months of operation. These are January, February and March.

Step 1: The structure of an optimal solution

Out first step in dynamic programming paradigm is to characterize the structure of an optimal solution.

We can think of each month in our problem as a stage in dynamic programming formulation.

In figure 3.1, stage 3 is January, stage 2 February and stage 1 March. The ending inventory of January is the beginning (X_2) of February and so on.

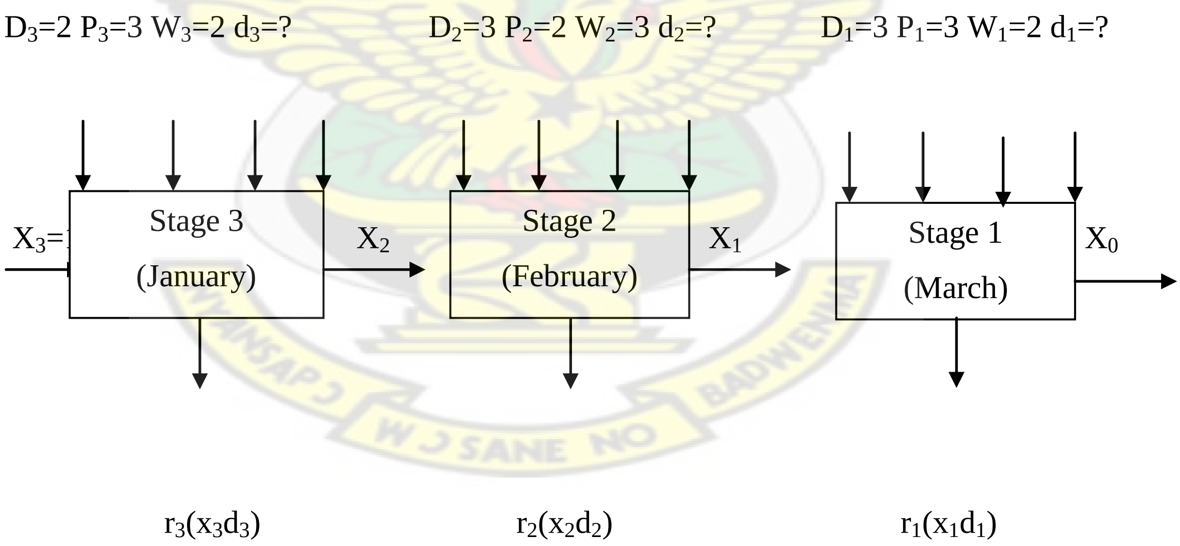


Figure 3.1: Schematic representation of the production and inventory control problem as a three – stage dynamic programming problem

(i) For state 3 (January)

We minimize the sum of production cost and inventory holding cost in the Month of January subject to demand (D_3=2), production capacity (P_3=3), warehouse capacity (W_3=2) and ending inventory, X_2. X_3 =1 is beginning inventory.

The stage transformation function for the month January is of the form.

$Ending\ inventory – beginning\ inventory\ (x1 = 1) +\ production\ (= 3)$

$demand\ (D3 = 2)$.

The return (objective) function for January is the sum of production and inventory holding costs in January and is given by $f3(x3)\ =\ min. r3(x3, d3)\ +\ f2\ (x2)$.

(ii) For stage 2 (February)

We minimize the sum of production cost and inventory holding cost in the Month of February subject to demand (D_2=3), production capacity (P_2=2), warehouse capacity (W_2=3) and ending inventory.

The stage transformation function for the month February is of the form:

$$Ending\ inventory\ = beginning\ inventory\ + production – demand$$

$$(D2 = 3), ie\ x1 = x2 + P2 - D2.$$

This shows that the solutions (x_2) for the previous period (January) is needed to find the current solution x_1.

The return function for February is the sum of production cost and inventory holding cost in February. The inventory holding cost depends partly on the ending inventory (x_2) of the previous period.

(iii) The stage 1 (March)

We minimize the sum of production cost and inventory holding cost in the of March subject to demand (D_1=3), production capacity (P_1=3), warehouse capacity (W_1=2) and ending inventory.

The stage transformation function for the month of March is of the form:

Ending inventory=beginning inventory + production-demand (D_1=3).

Return function for March is the sum of production and inventory holding costs in March.

Step 2: A recursive solution

In figure 1, we have numbered the periods backward; that is, stage 1 corresponds March, stage 2 corresponds to February and stage 3 corresponds to January. The stage transformation functions being the equation:

$$Ending\ inventory\ =\ beginning\ inventory\ +\ production - \ demand.$$

Thus, we have $x_n - 1 = x_n \ + d_n - D_n$

x_3=1 for the inventory beginning of January.

$x_2 = x_3 \ + \ d_3 - D_3 = x_3 + d_3 - 2$ for inventory ending January/beginning February.

$x_1 = x_2 \ + \ d_2 - D_2 = x_2 + d_2 - 3$ for inventory ending February /beginning March.

$x_0 = x_1 \ + \ d_1 - D_1 = x_1 + d_1 - 3$ for inventory ending March.

The return functions for each stage represent:

The sum of production and inventory holding costs for the month

i.e. $r_n(x_n, d_n) = C_n d_n + H_n (x_n + d_n - D_n)$.

i) For stage 1 : n = 1 (March)

r_1 (x_1m d_1) = 200d_1 + 40(x_1 + d_1-3) represents the total production and holding costs for the period. The production costs are $200 per unit and the holding costs are $40 per unit of ending inventory.

The other return functions are:

ii) For February $n = 2$

$$r_3(x_2, d_2) = 150d_2 + 30\,(x_2 + d_2 - 3) \quad stage\ 2$$

iii) For January $n = 3$,

$$r_3(x_3, d_3) = 175d_3 + 30\,(x_3 + d_3 - 2) \quad stage\ 3$$

There are three constraints that must be satisfied at each stage as we perform the optimization procedure. The first constraints is that the ending inventory must be less than or equal to the warehouse capacity. Mathematically we have

$$X_n + d_n - D_n \le W_n \ or\ x_n + d_n \le D_n + W_n \ \dots\dots\dots\dots\dots\ (1).$$

The second constraint is that the production level in each period must not exceed the production capacity. Mathematically we have

$$d_n \le p_n \dots\dots\dots\dots\dots (2)$$

For each stage, we must have the constraint that requires beginning inventory plus production to be greater than or equal to demand.

Mathematically this constraint can be written as

$$x_n + d_n \geq D_n \dots\dots\dots (3)$$

The inventory problem is then formulated as:

$$F_n(x_n) = min\{r_n(x_n, d_n) + f_{n-1}(x_{n-1})$$

Subject to

$x_n + d_n \leq D_n + W_n$

$x_n + d_n \leq D_n$

$d_n \leq P_n$

$x_n, d_n \geq 0$

where $f_{n-1}(x_{n-1})$ is the minimum value of the return function of x_{n-1}.

Step 3: Computing stage wise the optimal costs

i) Computations for stage 1 (March)

Our unknowns from the inventory problem are x_n, d_n. Since the problem is a discrete problem x_n, d_n are discrete. However, they should satisfy the constraints $x_n + d_n \leq D_n + W_n$

$d_n \leq P_n, x_n + d_n \geq D_n$

For stage 1 (March) we have n = 1, $D_1 = 3$, $P_1 = 3$, $W_1 = 2$, $C_1 = 200$, $H_1 = 40$.

From $d_1 \leq p_1 = 3, d_1 = 0, 1, 2, 3$ and $x_1 + d_1 \geq 3$ we get $x_1 = -$ 0,1,2,3. We use the values of x_1, d_1 to compute the minimum cost for stage 1.

Since we are attempting to minimize cost, we will want the decision variable d_1 to be as smaller as possible and still satisfy the demand constraint.

$$F1(x_1) = Min\{r_1(x_1, d_1) = 240d_1 + 4x_1 - 120$$

Subject to

$x_1 + d_1 \leq 5$ warehouse constraint,

$d_1 \leq 3$, production constraint and

$x_1 + d_1 \geq 3$ demand constraint.

$$x_1 = 0$$

$$f_1(0) = min\{240 \times 3 + 40 \times 0 - 120 = 600, d_1 = 3\}$$

Result $f_1(0) = 600$. Thus d_1

$x_1 = 1, d_1 = 2, d_1 = 3$

$$f_1(1) = min\begin{cases} 240 \times 2 \times 40 \times 1 - 120 = 400, d_1 = 2 \\ 240 \times 3 + 40 - 120 = 640, \quad d_1 = 3 \end{cases}$$

Result $f_1(1) = 400$. $Thus \ d_1^* = 2$

$x1 = 2, \qquad d1 = 1, d1 = 2, d1 = 3.$

$$f_1(2) = min\begin{cases} 240 \times 1 + 40 \times 2 - 120 = 200, & d_1 = 1 \\ 240 \times 2 + 80 - 120 = 440, & d_1 = 2 \\ 240 \times 3 + 80 - 120 = 680, & d_1 = 3 \end{cases}$$

Result $f_1(2) = 200$. Thus $d_1^* = 1$.

$x1 = 3, \quad d1 = 0, d1 = 1, d1 = 2.$

$$f_1(3) = min\begin{cases} 240 \times 0 + 40 \times 3 - 120 = 0, & d_1 = 0 \\ 240 \times 1 + 120 - 120 = 240, & d_1 = 2 \\ 240 \times 2 + 120 - 120 = 480, & d_1 = 3 \end{cases}$$

Result $f_1(3) = 0$. Thus $d_1^* = 0$.

Table 3.2 below contains x_1 *and* d_1 which take on values 0,1,2,3 and the values of $f_1(x_1)$. M is used to represents no feasible solution and the last column is the optimal solution.

Table 3.2 summary of results for March

		d_1				
		0	1	2	3	(d_1^*, f_1)
	0	M	M	M	600	(3,600)
	1	M	M	400	640	(2,400)
x_1	2	M	200	440	680	(1, 200)
	3	0	240	780	720	(0, 0)

We proceed to stage 2

ii) Computations for stage 2 (February)

$$f_2(x_2) = \min\{r_2(x_2, d_2) + f_1(x_1) = 150d_2 + 30(x_2 + d_2 - 3) + f_1(x_1)$$

$$= 180d_2 + 30x_2 - 90 + f_1(x_1)$$

Subject to

$x2 + d2 \leq 6, d2 \leq 2, x2 + d2 \geq 3$, and for each x2 selected we calculated

$x1 = x2 + d2 - 3$. Thus $d2 = 0, 1, 2$ and $x2 = 0, 1, 2, 3, 4$.

$f_2(x_2) = \min\{180d_2 + 30x_2 - 90 + f_1(x_1)$; $\mathrm{d2} = 0, 1, 2, \mathrm{x2} = 0, 1, 2, 3, 4$.

Note $x = 0$ is not feasible since 0 plus either 1 or 2 is not up to 3 and $x2 + d2 \geq 3$ is not satisfied.

$x2 = 1, \quad d2 = 2$ and $x1 = 1 + 2 - 3 = 0$

$f_2(1) = \min\{180 \times 1 \times +30 \times 1 - 90 + f_1(0) = 360 + 30 - 90 + 600 = 900, d_2 = 2$.

$f_2(1) = 900.$ $Thus$ $d_2^* = 2$.

$x_2 = 2$, $d_2 = 1$, $d_2 = 2$ and $x1 = 2 + 1 - 3 = 0$, $x1 = 1 + 2 - 3 = 0$ respectively

$$f_2(2) = min\begin{cases} 180 \times 1 + 30 \times 2 - 90 + f_1(0) = 180 + 60 - 90 + 600 = 750, d_1 = 1 \\ 180 \times 2 + 60 - 90 + f_1(1) = 360 + 60 - 90 + 400 = 730, \quad d_2 = 2 \end{cases}$$

$f_2(2) = 730$. Thus $d_2^* = 2$.

$x2 = 0, x2 = 1, x2 = 2$ and $d2 = 1, d2 = 2$

Table 3.3 below contains x_2 and d_2 which take on values 0, 1, 2 the values of $f_2(x_2)$ and the values of (d_2^*, f_2^*). M is used to represents no feasible solution.

Table 3.3 summary of results for February

		d_2			
		0	1	2	(d_2^*, f_2^*)
	0	M	M	M	-
x_2	1	M	M	900	(2,900)
	2	M	750	730	(2,730)

iii) Computations for stage3 (January)

$F_3(x_3)$=min {r3(x_3, d_3) + $f_2(x_2)$ = 150d_3 + 30(x_3 + d_3-3) + $f_2(x_2)$ = 180d_3 + 30x_3 – 90 + $f_2(x_2)$

$X_3 + d_3 \leq 4$, $d_3 \leq 3$, i.e d_3 = 1, 2, 3. $X_3 + d_3 \geq 2$. With x_1 = 1 already by the beginning inventory level and $x_2 = x_3 + d_3 - 2$.

$f_3(x_3)$ = min. {250d_3 + 30x_3 – 60 + f_2 (x_2).

x_3 = 1, d_3 = 1, 2, 3 and x_3 = 1 + 1 – 2 = 0, x_2 = 1 + 2 – 2 = 1, x_2 = 1 + 3 – 2 = 2 respectively

$$f_3(1) = \min \begin{cases} 205 \times 1 + 30 \quad \times \ 1 - 60 + \ f_2(0) = 175 + M, & d_1 \ = 1 \\ 205 \times 2 + 30 \ - 60 + \ f_2(1) = 380 + 900 \ = 1280, & d_3 \ = 1 \\ 205 \times 3 + \ 30 \ - 60 \ + \ f_2(2) = 615 - 30 + \ 730 \ = 1315 \ , & d_3 \ = 3. \end{cases}$$

Result $f_3(1) \ = \ 1280$. Thus $d_3^* \ = \ 2$.

Where $f_3(0)$ is not feasible and is denoted by M.

Table 3.4 below contains x_3 and d_3 which take on values 0, 1, 2, 3, the values of $f_3\ (x_3)$ and the values of (d_3^*, f_3^*). M is used to represents no feasible solution.

Table 3.4: Summary of results for January

		d_3			
		0	1	2	(d_3^*, f_3^*)
x_3	1	M	M	1280	2,1280

Thus, we find that the total cost assumed with the optimal production and inventory policy is 1280. The optimal solution is $f_3(1) = 380 + f_2(1) = 380 + 900 = 1280.$

Note $f_2(1)$ is obtained from table 3.3 as $f_2(1) = 300 + f_1(0) = 300 + 600 = 900$ where $f_1(0) = 600$ is also from table 3.2.

Step 4: Optimal solution from the computer results

To find the optimal decisions and inventory levels for each period, we may trace back through each stage and identify x_n and d_n^* as we go.

The company should produce two (2) units of cars with a beginning inventory one (1) in January of a production and inventory holding cost of \$380. Moreover, the company should produce two units of cars with a beginning inventory one (1) in February of a production and inventory holding of \$300 and three units of cars with a beginning inventory zero (0) in March of a production and inventory holding cost of \$600.

Table 3.5 summarizes the optimal production and inventory policy. In column one is the month, column two the beginning inventory, column three production cost, column four ending inventory, column five the holding cost and the last column total monthly cost.

Table 3.5: Summary of results for the optimal

Month	Beginning inventory	Production capacity(P_n)	Production cost(C_n d_n)	Ending inventory	Holding cost($H_n x_n$-1)	Total monthly cost
January	1	2	\$350	1	\$30	\$380
February	1	2	\$300	0	0	\$300
March	0	3	\$600	0	0	\$600
Total			\$1250		\$30	\$1280

3.5 THE STAGECOACH PROBLEM (NETWORK PROBLEM)

We consider a simple but illustrative deterministic dynamic programming problem that is known in the operations research literature as the "stagecoach problem." It deals with a hypothetical 19th-century stagecoach company that transports passengers from California to New York. Although the starting point (California) and the destination (New York) are fixed, the company can choose the intermediate **states** to visit in each **stage** of the trip. We assume that the trip is completed in four stages (legs) where stage 1 starts in California, stage 2 starts in one of three states in the Mountain Time Zone (say, Arizona, Utah or Montana), stage 3 starts in one of three states in the Central Time Zone (say, Oklahoma, Missouri or Iowa) and stage 4 starts in one of two states in the Eastern Time Zone (North Carolina or Ohio).When stage 4 ends, the stagecoach reaches New York, which is the final destination. Since in those days travel by stagecoach was rather dangerous because of attacks by roaming criminals, life insurance was offered to the travelling passengers. Naturally, the cost of the insurance **policy** was higher on those portions of the trip where there was more danger. The stagecoach company thus faced the problem of choosing a route that would be cheapest and thus safest for its passengers. The main problem of the traveler is how to find the safest routes and the cheapest cost of insurance policy in order to minimize cost.

We minimize the cost of insurance from state (A) to state (J) subject to the safety of the route. Current cost = immediate cost (Stag 2) + minimum future cost (stage n + 1).

Step 0: Variables definitions and data.

The figure 3.2 below shows the cost on the edges.

Level 1 2 3 4 5

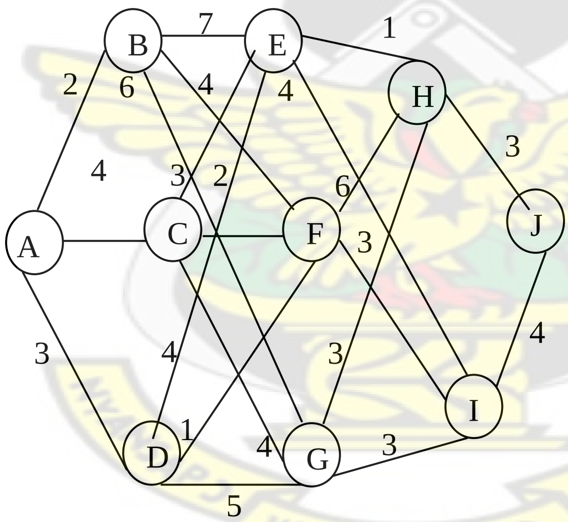


Figure 3.2: The road system and costs for the stagecoach problem

Table 3.6 showing the cost of insurance for moving from one state to another state and dash (-) is used to represent where cost is neither available nor applicable.

Table 3.6: The road system and costs for the stagecoach problem

	A	B	C	D	E	F	G	H	I	J
A	-	2	4	3	-	-	-	-	-	-
B	-	-	-	-	7	4	6	-	-	-
C	-	-	-	-	3	2	4	-	-	-
D	-	-	-	-	4	1	5	-	-	-
E	-	-	-	-	-	-	-	1	4	-
F	-	-	-	-	-	-	-	6	3	-
G	-	-	-	-	-	-	-	3	3	
H	-	-	-	-	-	-	-	-	-	3
I	-	-	-	-	-	-	-	-	-	4
J	-	-	-	-	-	-	-	-	-	-

Let us consider the cheapest possible ways to get from starting point (A) through to the last point (J).

Let $C_{x_{n,j}}$, $x_{n+1,j}$ denote the cost of insurance from stage n to stage n + 1.

Let I be the point it is at the stage n and j the route it should take.

Let $x_{n,i}$ be the state variable

Let $f_n(x_{n,i})$ denote the aluminum cost of the objective function from any city $x_{n,i}$to the final destination J.

Step 1: The structure of an optimal solution

The first step of dynamic programming is to characterize the structure of an optimal solution.

Let divide the problem into five stages as follows;

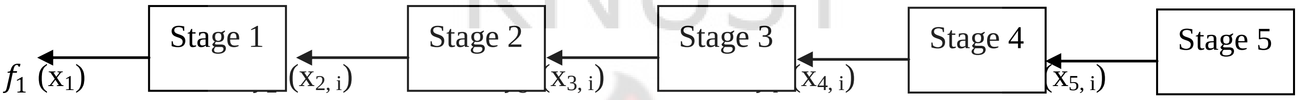


Figure 3.3: Schematic representation of the stagecoach problem as a five – stage dynamic programming problem.

Stage 1: Consists of city A

The return functions is $f_1(x_1)$ = immediate cost (stage 1) + minimum future cost (stag 2).

State 2: Consists of cities B, C, and D.

The return functions is $f_2(x_2)$ = immediate cost (stage 2) + minimum future cost (stag 3).

State 3: Consists of cities E, F, and G.

The return functions is

$f_3(x_3) \ = \ immediate\ cost\ (stage\ 3)\ +\ minimum\ future\ cost\ (stag\ 4).$

State 4: Consists of cities H and I.

The return functions is ：

$f_4(x_4) = \ immediate\ cost\ (stage\ 4)\ +\ minimum\ future\ cost\ (stag\ 5).$

State 5: Consists of cities J.

We have this as the final stage. Therefore $f_5(x_4)$

For the stagecoach problem, we start with the smaller problem where Mr. Ebenezer has nearly completed his journey and has only one more stage (stagecoach run) to go. The obvious optimal solution for this smaller problem is to go from his current state (whatever it is) to his ultimate destination (state J). For subsequent iterations, the problem is enlarged by increasing by 1 the number of stages left to go to complete the journey.

Step 2: A recursive solution

Let $f_n(x_{n,\,i})$ denote the optimal value of the objective function form any city $x_{2,\,i}$ to the final destination J. Hence the optimum is $f_1(x_1)$ the minimum of the sum of cost insurances from A to J.

Thus $f_n(x_{n,i}) = \min\{C_{x_{n,i},x_{n+1,j}} + f_{n+1}\left(x_{n+1,j}\right)$

Subject to $C_{x_{n,i},x_{n+1,j}} \geq 0$ and an integer.

i. For stage 1 : n = 1, x_1 = A

We minimize the cost of insurance from stage 1 (A) to stage 5 (J)

f_1 $(x_{n,\,i})$ = min $[C_{x_{1,1},x_{2,1}} + f_2\left(x_{2,1}\right), C_{x_{1,2},x_{2,2}} + f_2(x_{2,2})]$,. i = 1, and j = 1, 2, 3

ii. For stage 2 : n = 2

We minimize the cost of insurance from stage 2 (B, C, D) to stage 5(J)

f_2 $(x_{2,\,i})$ = min $[C_{x_{2,i},x_{3,j}} + f_3\ (x_{3,j})]$, i = 1, 2, 3 and j = 1, 2, 3.

iii. For stage 3 : n = 3

We minimize the cost of insurance from stage 3 (E, F, G) to stage 5(J)

f_3 ($x_{3,\,i}$) = min [$C_{x_{3,j},x_{4,j}}$ + f_4 ($x_{4,j}$)], i = 1, 2, 3 and j = 1, 2, 3.

iv. For stage 4 : n = 4

We minimize the cost of insurance from stage 4 (H, I) to stage 5(J).

f_4 ($x_{4,\,i}$) = min [$C_{x_{4,j},x_{5,j}}$ + f_5 ($x_{5,j}$)], i = 1 and j = 1, 2.

iv) For stage 5 : n = 5, x_5 = J.

Since the ultimate destination (state J = x_5) is reached at the end of stage 5, $f_5(x_5)$ = 0.

Let $f_n(x_{n,i})$ be the optimal value of the objective function form any stage n to the final destination J.

Step 3: Computing the stage wise optimal cost

i) Computations for stage 5(x_5 = J): n = 5.

Since J is the final stage there is no cost after J, then $f_5(x_5)$ = 0

ii) Computations for stage 4($x_{4,1,}$ = H, $x_{4,2}$=I): n = 4, I = 1, 2.

When he has only one more stage to go (n = 4), the best route is determined entirely by his current state (either H or 1) and his final destination x_5 = J.

Result $f_4\left(x_{4,1}\right) = f_4(H) = C_{x_{4,1}x_{5,1}}$+ $f_5(x_{5,1})$ = 3 + 0 = 3.

Result $f_4(x_{4,2})$ = f_4(I) = $C_{x_{4,2}x_{5,2}}$ + $f_5(x_5)$ = 4 + 0 = 4

ii) Computations for stage 3(x_{31} = E, $x_{3,2}$ = F and $x_{3,3}$ = G): n = 3, i = 1,2.

When he has two more stages to go (n=3).

$$f_3(x_{3,1}) = min\begin{cases} C_{x_{3,1},x_{4,1}} + f_4(x_{4,1}) = 1 + 3 = 4 \\ C_{x_{3,2},x_{4,2}} + f_4(x_{4,2}) = 4 + 4 = 8 \end{cases}$$

Result $f_3(x_{3,1})$ = 4

$$f_3(x_{3,2}) = min\begin{cases} C_{x_{3,1},x_{4,1}} + f_4(x_{4,1}) = 6 + 3 = 9 \\ C_{x_{3,2},x_{4,2}} + f_4(x_{4,2}) = 3 + 4 = 7 \end{cases}$$

Result $f_3(x_{3,2})$ = 7

$$f_3(x_{3,3}) = min\begin{cases} C_{x_{3,1},x_{4,1}} + f_4(x_{4,1}) = 3 + 3 = 6 \\ C_{x_{3,2},x_{4,2}} + f_4(x_{4,2}) = 3 + 4 = 7 \end{cases}$$

Result $f_3(x_{3,3})$ = 6

iii) Computations for stage $2(x_{2,1} = B, x_{2,2} = C\ and\ x_{2,3} = D,): n\ =\ 2$

The solution for the second stage problem (n = 2), where there are three stages to go.

$$\mathrm{f_3(x_{3,3}) = min} \begin{cases} \mathrm{C_{x_{2,1},x_{3,1}} + f_3(x_{3,1}) = 7 + 4 = 11} \\ \mathrm{C_{x_{2,2},x_{3,2}} + f_3(x_{3,2}) = 4 + 7 = 11} \\ \mathrm{C_{x_{2,3},x_{3,3}} + f_3(x_{3,3}) = 6 + 6 = 12} \end{cases}$$

$result\ f_2(x_{2,1})$ = 11

$$\mathrm{f_2(x_{2,2}) = min} \begin{cases} \mathrm{C_{x_{2,1},x_{3,1}} + f_3(x_{3,1}) = 3 + 4 = 7} \\ \mathrm{C_{x_{2,2},x_{3,2}} + f_3(x_{3,2}) = 7 + 7 = 9} \\ \mathrm{C_{x_{2,3},x_{3,3}} + f_3(x_{3,3}) = 4 + 6 = 10} \end{cases}$$

$result\ f_2(x_{2,2})$ = 7

$$f_2(x_{2,3}) = \min \begin{cases} C_{x_{2,1},x_{3,1}} + f_3(x_{3,1}) = 4 + 4 = 8 \\ C_{x_{2,2},x_{3,2}} + f_3(x_{3,2}) = 7 + 7 = 8 \\ C_{x_{2,3},x_{3,3}} + f_3(x_{3,3}) = 6 + 6 = 12 \end{cases}$$

$result\ f_2(x_{2,3})$ = 8

i) Computations for stage 1 (x_1 = A)

Moving to the first – stage problem (n = 1), with all four stages to go.

$$f_1(x_1) = \min \begin{cases} C_{x_{1,1},x_{2,1}} + f_2(x_{2,1}) = 2 + 11 = 13 \\ C_{x_{1,2},x_{2,2}} + f_2(x_{2,2}) = 4 + 7 = 11 \\ C_{x_{1,3},x_{2,3}} + f_2(x_{2,3}) = 3 + 8 = 11 \end{cases}$$

$Result\ f_1(x_1)$ = 11

Since 11 is the minimum cost, $f_1(A)\ = 11\ and\ x_{2,2} = C\ or\ x_{2,3} = D$

Step 4: Optimal solution from the computed results

An optimal solution for entire problem can now be identified from the results above. Results for the n = 1 problem that Mr. Ebenezer should go initially to either state C or state D. suppose that he chooses $x_{2,2}$ = C. For n = 2, the results for $x_{3,1}$ = E. This result leads to the n = 3, which gives $x_{4,1}$ = H for $x_{3,1}$ = E and the n = 4 yields x_5 = J $x_{4,1}$ = H.

Hence, one optimal route is A → C → E → H → J. Choosing $x_{2,2}$= D leads to the other two optimal routes A → D → E → H → J and A → D → F → I → J. They all yield a total cost of f_1 (A) = 11. These results of dynamic programming analysis also are summarized in the diagram below (Hillier et al, 2005). In graphical display of the dynamic programming solution of the stagecoach problem, each arrow shows an optimal policy decision (the best immediate destination) from that state, where the number b the resulting cost from there to the end is shown in figure 3.4.

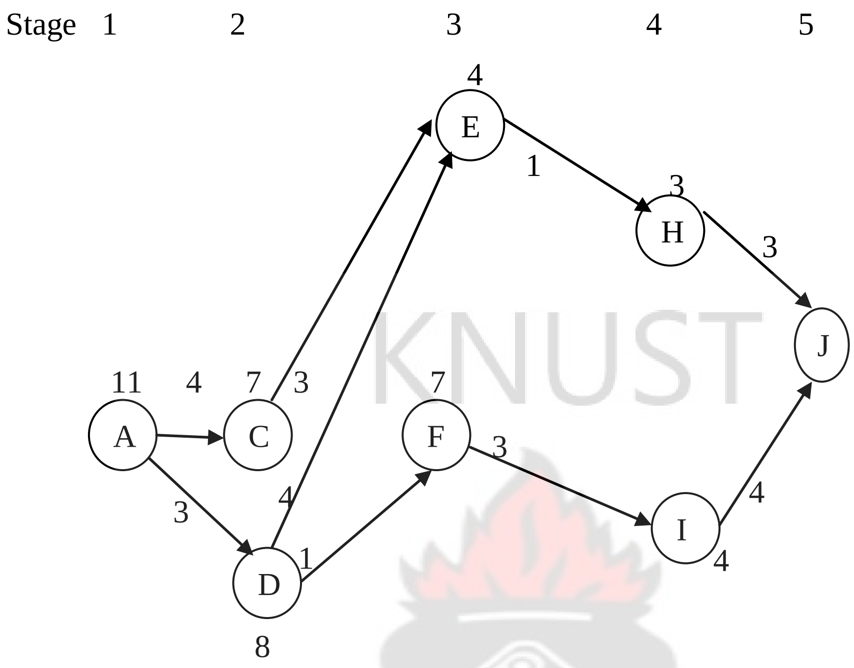


Figure 3.4: A representation of the optimal solution of the stagecoach problem

3.6 THE KNAPSACK PROBLEM

We consider maximizing benefit (total value rating) subject to the number of days available (10) for processing of a job and the number of jobs available. The stage transformation functions are then defined as:

x_{n-1} = the number of days available at stage n – the product of the number days needed to compete one job by the number of jobs to process.

The return functions at each stage are based on the value rating of a job times the number of jobs selected for processing.

The first constraints is that the number of days needed to process a job must be less than or equal to the number days available (10).

Secondly, the number of jobs selected must be less than or equal to the number of jobs available. The basic idea of the Knapsack problem is that there are N different types of items that can be put into a knapsack. Each item has a certain weight associated with it as well as a value. The problem is to determine how many units each item to place in the knapsack in order to maximum total value. A constraint is placed on the maximum weight permissible.

We consider a manager of a manufacturing operation who has selection of jobs to process during the following 10 – day period. A list of the jobs waiting to be processed at the beginning of the current week is presented in table 3.7. The estimated time required for completion and the value rating associated with each category of job are also shown in the table. The main aim of the Manager is to find out how many jobs to choose from each category to process in order to maximize performance value, David, et al (1988) indicates.

Step 0 Variable definitions and data

Table 3.7: Job data for the manufacturing operation

Job number n	No. of jobs to be processed (N).	Estimated completion time per job (days (u_n).	Value rating
Category 1	4	1	2
Category 2	3	3	8
Category 3	2	4	11
Category 4	2	7	20

The value rating assigned to each job is a subjective score assigned by the supervisor. A scale from 1 to 20 is used to measure the value of each job, where 1 represents jobs of the least value, and 20 represents jobs most value. We would like to make a selection of jobs to process during the next 10 – days such that all the jobs selected can be processed in 10 days and that the total performance value of jobs selected is maximized. In knapsack problem terminology we are in essence selecting the best jobs for our 10-days knapsack, where the knapsack has a capacity equal to the 10-day (w) production capacity. We formulate and solve this problem using a dynamic programming solution procedure.

Let d_n denote the number of jobs in category n selected (that is, the decision variable at stage n). The state variable x_n ($x_n \leq$ w) is defined as the number of days of processing time remaining when we reach stage n.

d_n=decision variable, x_n = state variable, u_n = the number of needed to complete one job.

Step 1: The structure of an optimal solution

This problem can be formulated as a dynamic programming problem involving four stages.

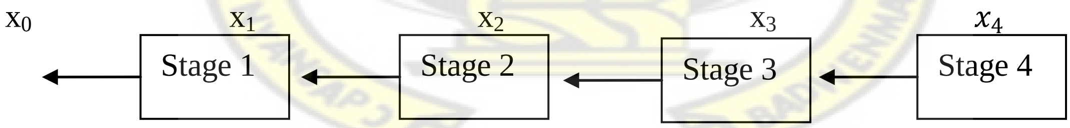


Figure 3.5: schematic representation of the knapsack problem as a four – stage dynamic programming problem.

Stage 1: Consists of category 1. The number of jobs to be selected for processing in category 1 should be less than or equal to 4.

The stage transformation functions are then defined as:

$$x_0 = t_1(x_1, d_1) = x_1 - u_1 d_1. \quad d_1 \leq N$$

The return function is $r_1(x_1, d_1) = 2d_1, \; d_1 \leq 4$

Stage 2: Consists of category 2. The number of jobs to be selected for processing in category 2 should be less than or equal to three (3).

The stage transformation functions are then defined as:

$$x_1 = t_1\,(x_1, d_1)x_2 - u_2 d_2. \quad d2 \leq N2$$

The return function is $f_1(x_2) = r_2(x_2, d_2) + 8d_2 + f_1(x_1), \quad d2 \leq 3$

Stage 3: Consists of category 3. The number of jobs to be selected for processing in category 3 should be less than or equal to two (2).

The stage transformation functions are then defined as:

$$x_2 = t_3(x_3, d_3) = x_3 - u_3 d_3\,. \quad d_3 \leq N_3$$

The return function is $f_3(x_3) = r_3\,(x_3, d_3) + f_2\,(x_2) = 11d_3 + f_2(x_2)$

Stage 4: Consists of category 4.

The number of jobs to be selected for processing in category 4 should be less than or equal to two (2). The stage transformation functions are then defined as:

$$x_3 = t_4(x_4, d_4) = x_4 - u_4 d_4. \quad d_{\,4} \leq N_4$$

The return function is

$$f_4(x_4) = r_4(x_4, d_4) + f_3(x_3) = 20d_4 + f_3(x_3)$$

At stage 1 we must decide how many jobs from category 1 to process, at stage 2 we must decide how many jobs from category 2 to process, and so on. Thus we let d_n denote the number of jobs in category n selected (that is, the decision variable at stage n). The state variable $x_n(x_n \leq w)$ is defined as the number of days of processing time remaining when we reach stage n.

Stage 2: A recursive solution

Thus with a 10 – day production period, $x_4 = 10$ represents the total number of days that are available for processing jobs. The stage transformation functions are then defined as:

$X_{n-1} = t_n(x_n, d_n) = x_n - u_n d_n. \quad d_n \leq N_n.$

$f_n(x_n) = r_n(x_n, d_n) + f_{n-1}(x_{n-1}).$

The $f_n(x_n)$ is the total return function after decision d_n is made.

i. Stage 4 : n = 4, $x_3 = t_4(x_4, d_4) = x_4 - 7d_4$

The return at each stage is based on the value rating of jobs and the number of jobs selected at each stage. The return functions are as follows:

$r_4(x_4, d_4) = 20d_4. \quad d_4 \leq 2$

$f_n(x_n) = r_n(x_n, d_n) + f_{n-1}(x_{n-1}).$

The $f_n(x_n)$ is the total return function after decision d_n is made.

$f_4(x_4) = r_4(x_4, d_4) + f_3(x_3) = 20d_4 + f_3(x_3).$

ii. For stage 3 : n = 3

$x_2 = t_3(x_3, d_3) = x_3 - 4d_3$.

$r_3(x_3, d_3) = 11d_3$. $d_3 \leq 2$.

$f_3(x_3) = r_3(x_3, d_3) + f_2(x_2) = 11d_3 + f_2(x_2)$

iii. For stage 2 : n = 2

$x1 = t2(x_2, d_2) = x2 - 3d3. r2(x_2, d_2) = 8d2$

$f_2(x_2) = r_2(x_2, d_2) + f_1(x_1) = 8d_2\ f_1(x_1)$, $d_2 \leq 3$

iv. For stage 1 : n = 1

$X_0 = t_1(x_1, d_1) = x_1 - d_2$. $r_1(x_1, d_1) = 2d_1$, $d_1 \leq 4$

Step 3: Computing the cheapest cost at each category

We will apply a backward solution procedure; that is, we will begin by considering the stage 1 decision.

i) Computations for stage 1 (filling with item category 1 only): n = 1.

Note that the input to stage 1, x_1, which is the number of days of professing time available at stage 1, is unknown because we have not yet identified the decisions at eh previous stages. Therefore in our analysis at stage 1 we will have to consider all possible valleys x_1 and identify the best decision d_1 fir each case; $f_1(x_1)$ will be the total return after decision d_1 is made. $f_1(x_1) = r_1(x_1, d_1) = 2d_1$. $d_1 \leq d$ also we are to consider all possible values of d_1 (that is 0, 1, 2, 3, or 4).

The number of category 1 jobs selected will depend upon the processing time available but cannot exceed 4.

Recall that $f_1(x_1)$ represents the value of the optimal return from stage 1 and all remaining stages, given an input of x_1 to stage 1. Let us move on to stage 2 and carry out the optimization at that stage.

ii) For stage 2 (filling with items 1 and 2) : n = 2.

Since the input to stage 2, x_2, is unknown, we have to consider all possible values from 0 to 10. Also we consider all possible values of d_2 (that is, 0, 1, 2, or 3). $f2(x2) = 8d_2+f_1(x_1)$, $d_2 \leq 3$.

Results

Note that some combinations of x_2 and d_2 are not feasible. For example with x_2 =2 days, d_2=1 is feasible (i.e not possible) because category 2 jobs each job require 3 days to process.

iii) For stage 3 (filling 1, 2 and 3 items) : n=3.

$f\ (x_3)= 11d_3+f_2\ (x_2)$, $d_3 \leq 2$.

iv) Computations for stage 4(filling with 1,2,3 and 4): n = 4

$F_4 = 20d_4 + f_3\ (x_3)$, $d_4 \leq 2$.

The optimal solution is $f4\,(10)\ =\ 20\ +\ f3\,(3)\ =\ 28$. Note that $f3(3) = 0 + f2(3) = 8\ i.e\ f2(3) = 8x1 = 8$

Step 4: An optimal solution from the computed results

The optimal decision, given $x_4 = 10$, is $d^*_4=1$. In order to identify the overall optimal solution, we must now trace back through the tables beginning at stage 4. The optimal decision at stage 4 is $d^*_4 = 1$. Thus $x_3 = 10 – 7 =3$, and we enter stage 3 with 3 days available for processing. With $x_3 = 3$ we see that the best decisions at stage 3 is $d^*_3 = 0$. Thus, we enter stage 2 with $x_2 = 3$. The optimal decision at stage 2 with $x_2 = 3$ is $d^*_2 = 1$, resulting in $x_1=0$. Finally the decision at table 1 must be $d^*_1=0$. The table 3.8 below is the optimal solution of the number of jobs to be processed form each category.

Table 3.8 Summary of the optimal solution of the knapsack problem

Decision	Return
$d^*_1=0$	0
$d^*_2=1$	8
$d^*_3=0$	0
$d^*_4=0$	20
Total return	28

We should schedule one job form category 2 and one job from category 4 for processing over the next 10 -day planning period.

3.7 THE EQUIPMENT-REPLACEMENT PROBLEM WITH TRADE-IN COST

A replacement policy is a specification of a sequence of “keep” or “replace” actions, one for each period. Two simple examples are the policy of replacing the car every year and the policy of keeping the first car until the end of period N. An optimal policy is a policy that achieves the smallest total net cost of ownership over the entire planning horizon.

We consider a car which has to be operated throughout a planning horizon of N periods and when it reaches a specific age i will be more economical to replace. Given that each period corresponds to one year; and that we are required to make a decision as to whether or not to replace the car at the beginning of every year. The problem of interest is to determine an optimal replacement policy. Let:

$c(i) =$ The annual operating cost of an i-year-old car , $where\ i = 1, 2, \dots, N$.

p = The price of a new car.

$t(i) =$ The trade-in value of an i-year-old car , $for\ i = 1, 2, \dots, N$.

$S(i) =$ The salvage value of an i-year-old car at the end of year N , $for\ i = 1, 2, \dots, N$.

$i =$ State of the car i.e. age of the car at a given stage.

$k =$ Stage of the i.e. year

We derive the optimal policy for this problem using dynamic programming by organizing the solution procedure into four steps:

1. Definition of appropriate stages and states.

2. Definition of the optimal-value function.

3. Construction of a recurrence relation.

4. Recursive Computation.

Stages and States

Since we consider one decision per year, it is natural to make each year a stage. We shall refer to the year count (or index) as the *stage variable*. The definition of states requires a little bit more thought. It is worth noting that the state information corresponds to a specification of "where we are" within a given stage. We shall also refer to the age of the car in service at the beginning of a year as the *state variable*.

Optimal-Value Function

The optimal-value function is a function that returns, for any given pair of stage and state, the best possible total cost from that point to the end. With the stage and state variables appropriately defined, we define the optimal-value function as

$V_k(i) =$ the minimal total net cost from year k to the end of year N, starting with an i-year-old car in year k.

Recurrence Relation

We consider being at the beginning of year k with an i-year-old car and being reduced to only two available actions: keep or replace (the car).

For a given keep the i-year-old car action chosen, the immediate one-stage cost is simply $c(i)$. Since the next stage and state as a result of this action is $k + 1$ and $i + 1$, the minimal total future net cost from that point to the end is, by definition, $V_{k+1}(i + 1)$.

It follows that the best possible total net cost associated with the keep action is given by

$$c(i) + V_{k+1}(i + 1).$$

On the other hand, if the action chosen is to replace the i-year-old car. Then, the immediate one-stage cost is the sum of: p (the price of a new car), $-t(i)$ (the negative of the revenue from trading in the i-year-old car), and $c(0)$ (the operating cost of a new car).

Since the next stage and state as a result of this action is $k + 1$ and 1, the minimal total future net cost from that point to the end is, by definition $V_{k+1}(1)$. It follows that the best possible total net cost associated with the replace action is given by

$p - t(i) + c(0) + V_{k+1}(1)$. Since our goal is to minimize the total net cost, the recurrence relation is:

$$V_k(i) = min\,[c(i) + V_k + 1(i + 1), p - t(i) + c(0) + V_{k+1}(1)]\,. \qquad (1)$$

In general, it is arguable that the price of a new car should depend on the time period. Consequently, it may be desirable to replace p by a set of p_k's, where p_k is the price of a new car in year k. Such a scenario can be easily accommodated in our solution procedure by revising the recurrence relation to:

$$V_k(i) = min\,[c(i) + V_{k+1}(i + 1), p_k - t(i) + c(0) + V_{k+1}(1)]\,. \qquad (2)$$

Computation

With the recurrence relation in place, the final step of the solution procedure consists of the recursive computation of the $V_k(i)$'s.

ILLUSTRATIVE EXAMPLE

Suppose a 2-year old car is needed for three years. The annual cost of operating a car is a function of its age; and this cost function is given by:$c(0) = 10$, $c(1) = 20$, $c(2)40$, $c(3) = 60, and\ c(4) = 70$. The price of a new car is 60, i.e., p = 60. The trade-in value of a used car is a function of its age at the time of trade in; and this function is given by: $t(1) = 30, t(2) = 20, t(3) = 15, and\ t(4) = 10.$ Finally, the salvage value of a used car is again a function of its' age; and this function is given by: $s(1) = 20, s(2) = 15, s(3) = 10, s(4) = 0, and\ s(5) = 0.$ We determine an optimal replacement policy under the above given assumptions.

Stage and State

Since the car is needed for three (3) years, we have $N = 3$ and a 2-year old car at the beginning of the first year. Hence $1 \leq k \leq N$ where k shall be refer to the year count (or index) as the *stage variable*. We again refer to the age of a car in service at any given stage as i the *state variable*. Hence we shall always have an i-year old car at stage k to begin with.

Computation

We specify the boundary condition. For this purpose, it is convenient to view the end of year 3 as the beginning of a final stage 4, where the only available action is to salvage the car in service. Since the revenue received from salvaging a car can be interpreted as a negative cost, this yields the boundary condition specified in the table 3.9 with column 1 representing the various ages of the car at stage 4 and column 2 indicates the salvage values at various states.

Table 3.9: Solution Stage (4)

i	$V_4(i)$
1	-20
2	-15
3	-10
4	0
5	0

Note that the highest possible state is 5. This is a consequence of the fact that we begin year 1 with a 2-year-old car and the planning horizon is 3 years. We consider stage 3, where the highest possible state is 4. For state 1, the one-stage costs associated with the keep and replace actions are $c(1) = 20 \, and \, p - t(1) + c(0) = 60 - 30 + 10 = 40$ respectively. For state 2, the one-stage costs associated with the keep and replace actions are $c(2) = 40$ and $p - t(2) + c(0) = 60 - 20 + 10 = 50$, respectively. For state 3, the one-stage costs associated with the keep and replace actions are $c(3) = 60$ and $p - t(3) + c(0) = 60 - 15 + 10 = 55$ respectively. Finally, for state 4, the one-stage costs associated with the keep and replace actions are $c(4) = 70$ and $p - t(4) + c(0) = 60 - 10 + 10 = 60$, respectively. Substitution of these one-stage costs and the relevant $V_4(i)$'s from the stage-4 table above into the recurrence relation;

$$V_3(i) = min\,[c(i) + V_4(i + 1), p - t(i) + c(0) + V_4(1)]$$

Now yields the solution at stage 3 as seen in table 3.10 with column 1 representing the various ages of the car, columns 2 and 3 captures the costs in line with keep or replace actions respectively whereas columns 4 and 5 depict the total recursive cost and optimal decision associated with each state (age) of the car.

Table 3.10: Solution to stages (3):

	Action			
	Keep	Replace	$V_3(i)$	Optimal Action
1	20 + (−15) = 5	40 + (−20) = 20	5	Keep
2	40 + (−10) = 30	50 + (−20) = 30	30	Keep or Replace
3	60 + 0 = 60	55 + (−20) = 35	35	Replace
4	70 + 0 = 70	60 + (−20) = 40	40	Replace

Note that for state 2, the costs associated with the keep and replace actions are tied at 30; therefore, both actions are optimal. Next, we move back one more stage to stage 2, where the highest possible state is 3. For all three states, the one-stage costs associated with the keep and replace actions are identical to the ones computed earlier in stage 3. Substitution of these one-stage costs and the relevant $V_3(i)'s$ from the stage-3 table above into the recurrence relation

$V_2(i) = min\,[c(i) + V_3(i + 1), p - t(i) + c(0) + V_3(1)]$ yields the solution at stage 2 as shown in table 3.11 with column 1 representing the various ages of the car, columns 2 and 3 captures the costs in line with keep or replace actions respectively whereas columns 4 and 5 depict the total recursive cost and optimal decision associated with each state (age) of the car.

Table 3.11: Solution to stages (2)

i	$Actions$			
	Keep	Replace	$V_2(i)$	Optimal Action
1	20 + 30 = 50	40 + 5 = 45	45	Replace
2	40 + 35 = 75	50 + 5 = 55	55	Replace
3	60 + 40 = 100	55 + 5 = 60	60	Replace

It follows that we should replace the car in service regardless which state we happen to be in within this stage. Finally, in stage 1, the only state is 2. Substitution of $c(2) = 40, p - t(2) + c(0) = 60 - 20 + 10 = 50, V2(1) = 45, and\ V2(3) = 60$ into the recurrence relation

$V_1(2) = min\,[c(2) + V_2(3), p - t(2) + c(0) + V_2(1)]$ yields solution at stage 1 as evidenced in table 3.12 with column 1 representing the various ages of the car, columns 2 and 3 indicating the net costs in line with keep or replace actions respectively whereas columns 4 and 5 depict the total recursive cost and optimal decision associated with a $2 - year\ old\ car$ car.

Table 3.12: solution to stages (1)

	Actions			
i	Keep	Replace	$V_1(i)$	Optimal Action
2	40 + 60 = 100	50 + 45 = 95	95	Replace

Since $V_1(2) = 95$, we conclude that the minimal total net cost from year 1 to the end of year 3, starting with a 2-year-old car in year 1, is 95.

The sequence of optimal actions can be read from the above tables sequentially as follows.

An inspection of the stage-1 table shows that we should immediately replace the original 2-year-old car. This implies that the age of the car in service at the start of year 2 will be 1. Upon inspecting the first row of the stage-2 table, we see that we should replace again in year 2. Finally, from the first row of the stage-3 table, it is seen that we should keep the 1-year-old car at the start of year 3. Thus, the optimal policy prescribes the following sequence of actions: replace, replace, and keep. In summary, the total optimal net cost for this optimal policy {R, R, K} equals 95.

3.8 AUTOMOBILE REPLACEMENT PROBLEM WITH OR WITHOUT INCOME

The D.P. recursive equation of an automobile replacement problem for either Keep or Replace decision with the aim of minimizing the total cost can be written as in equation (3) if the organization fleet of buses generates some income. Equation (4) represents the D.P.

recursive equation with the aim of maximizing the company pure profit. But equation (5) represents the D.P. recursive equation for minimizing the total cost if there is no income

$$V_k(i) = min \begin{cases} C_k(i) - I_k(i) + V_{k+1}(i+1) & Keep \\ C_k(0) - I_k(0) + R_k(i) + V_{k+1}(1) & Replace \end{cases} \quad (3)$$

$$V_k(i) = max \begin{cases} I_k(i) - C_k(i) + V_{k+1}(i+1) & Keep \\ I_k(0) - C_k(0) + R_k(i) + V_{k+1}(1) & Replace \end{cases} \quad (4)$$

$$V_k(i) = min \begin{cases} C_k(i) + V_{k+1}(i+1) & Keep \\ C_k(0) + R_k(i) + V_{k+1}(1) & Replace \end{cases} \quad (5)$$

Where :

$C_k(i) =$ Represent total cost at each stage (k) of an old bus.

$C_k(0) =$ Represent total cost at each stage (k) of a new bus.

$I_k(i) =$ Represent the old bus income at stage (k).

$I_k(0) =$ Represent the new bus income at stage (k).

$R_k(i) =$ Represent the bus replacement cost at stage (k).

$V_k(i) =$ Represent the total recursive cost for a bus of age (i) at stage (k).

$V_{k+1}(i+1) =$ Represent the total recursive cost for a bus of age $(i+1)$ at stage $(k+1)$.

$V_{k+1}(1) =$ Represent the total recursive cost for a bus of age (1) at stage $(k+1)$

$i =$ Represent the bus age at stage k, (The state variable)

$D_k =$ Represent the decision at stage k.

$k =$ Represent the stage.

ILLUSTRATIVE EXAMPLE

We consider a 2-year old equipment with its data available in Tables 3.6.1 to 3.6.4 with row 1 clearly specifying the stage in question, row 2 indicates the state variables at a given stage, row 3 represents the old bus income at any given state with row 4 representing the replacement cost of the equipment in a given state at various stages, (all values in dollars). It is required to find the optimal replacement policy for this equipment to minimize the total cost over the next 4 years.

Table 3.6.1: Data of the illustrative example, stage 1

Stage	1	
i	0	2
$I_k(i)$	3000	2200
$C_k(i)$	1100	2800
$R_k(i)$	---	6200

Table 3.6.2: Data of the illustrative example, stage 2

Stage	2		
i	0	1	3
$I_k(i)$	5000	4600	3700

$C_k(i)$	1200	2450	6100
$R_k(i)$	---	5600	8000

TABLE 3.6.3: Data of the illustrative example, stage 3

Stage	3			
i	0	1	2	4
$I_k(i)$	7000	4800	4600	2700
$C_k(i)$	2300	2500	4000	6000
$R_k(i)$	---	5700	7500	8200

Table 3.6.4: Data of the illustrative example, stage 4

Stage	4				
i	0	1	2	3	5
$I_k(i)$	6800	5000	4700	4000	2500
$C_k(i)$	2400	2600	4100	5300	6600
$R_k(i)$	---	7900	6600	7200	8300

The decision will be taken at the beginning of each year. The problem will be solved by backward dynamic programming by using the recursive equation (1). The problem state variable is shown in Table *3.6.5* with row 1 representing the individual stages and row 2 identifying the state variables at various stages:

Table 3.6.5: State variables for 2 years old equipment

k	1	2	3	4
i	$i = 2$	$i = 1,3$	$i = 1,2,4$	$i = 1,2,3,5$

Table 3.6.6 summarizes the results obtained in various states at stages (4) with column 1 indicating the age of the equipment (state), columns 2 and 3 represent the costs associated with keeping and replacing an $i - year\ old$ equipment respectively. The last column stipulates the optimal decision to keep or replace at various states.

Table 3.6.6: Solution of stages (4)

i	Keep	Replace	$V_4(i)$	D_4
5	4100	3900	3900	Replace
3	1300	2800	1300	Keep
2	-600	2200	-600	Keep
1	-2400	3500	-2400	Keep

Table 3.6.7 summarizes the results obtained in various states at stages (3) with column 1 indicating the age of the equipment (state), columns 2 and 3 represent the costs associated with keeping and replacing an $i - year\ old$ equipment respectively. The last column stipulates the optimal decision to keep or replace at various states.

Tables 3.6.7: Solution of stages (3)

i	Keep	Replace	$V_3(i)$	D_3
4	7200	1100	1100	Replace
2	700	400	400	Replace
1	-2900	-1400	-2900	Keep

Table 3.6.8 summarizes the results obtained in various states at stages (2) with column 1 indicating the age of the equipment (state), columns 2 and 3 represent the costs associated with keeping and replacing an $i - year\ old$ equipment respectively. The last column stipulates the optimal decision to keep or replace at various states.

Tables 3.6.8: Solution of stages (2)

i	Keep	Replace	$V_2(i)$	D_2
3	3500	1300	1300	Replace
1	-1700	-1100	-1750	Keep

Table 3.6.9 summarizes the results obtained in various states at stages (1) with column 1 indicating the age of the equipment (state), columns 2 and 3 represent the costs associated with keeping and replacing a $2-year\ old$ equipment respectively. The last column stipulates the optimal decision to keep or replace at state 2.

Tables 3.6.9: Solution of stages (1)

i	Keep	Replace	$V_1(i)$	D_1
2	1900	2550	1900	Keep

From Tables 3.6.6 to 3.6.9, we obtain the optimal replacement policy in the backward movement fashion as shown in Table 3.6.10 with row 1 and 2 specifying the stage and keep or replace decisions respectively. The last column spells out the total cost in pursuance of the outlined policy.

Table 3.6.10: The optimal replacement policy and its total cost

Stage	1	2	3	4	Total Cost
Decision	Keep	Replace	Keep	Keep	$1,900

It is evidenced that the company should keep the equipment at the first year, then replaces it by a new one, then keep the new equipment till the rest of the planned period. The total optimal cost for this optimal policy {K, R, K, K} equals $1900.

CHAPTER FOUR

DATA COLLECTION, ANALYSIS AND RESULTS

4.1 DATA COLLECTION

Metro-Mass Transit Ltd (MMT) is a passenger transport company in Ghana with enviable track record. This company has a fleet size of more than 300 different buses with the Kumasi depot having 94 buses of four distinct types.

Our research study is carried out on 5 buses, a bus each of the kinds available, namely VDL Daf, VDL Jonckheere, VDL Commuter, VDL Neoplan City (1^{st} generation) and VDL Neoplan City (2^{nd} generation).

The studied planned period is 11 years which starts from the year 2006 to 2016. The actual data are collected for the years 2006 to 2010. Then Microsoft Excel is used to predict the future values for the rest of the planned period. The tables (first three columns) in appendix ‘A’ to ‘E’ represent our case study collected and predicted data for the buses. The collected data include: the types of buses, replacement cost of buses, maintenance cost of buses and income generated (yr) by each bus for the planned period years. The income generated by the bus in most cases decrease with increasing age of the bus at any given stage whiles the cost of maintaining it increases with increasing age of the bus.

4.1.1 MODEL FORMULATION

The decision will be taken at the beginning of each year. The problem will be solved by backward dynamic programming using the recursive equation 4 (shown in chapter three).

The problem is to find the maximum net profit in operating each bus over the planned period. The problem is formulated as a dynamic programming problem with the assumption that a bus can only be kept or replaced at the beginning of each year. The bus is again not subjected to catastrophic failure. The mathematical notation and formulation are as follows:

Let

$C_k(i)$ = Represent total cost at each stage (k) of an old bus.

$C_k(0)$ = Represent total cost at each stage (k) of a new bus.

$I_k(i)$ = Represent the old bus income at stage (k).

$I_k(0)$ = Represent the new bus income at stage (k).

$R_k(i)$ = Represent the bus replacement cost at stage (k).

$V_k(i)$ = Represent the total recursive net profit for a bus of age (i) at stage (k).

$V_{k+1}(i+1)$ = Represent the total recursive net profit for a bus of age $(i+1)$ at stage $(k+1)$.

$V_{k+1}(1)$ = Represent the total recursive net profit for a bus of age (1) at stage $(k+1)$

i = Represent the bus age at stage k, (The state variable)

D_k = Represent the decision at stage k.

k = Represent the stage.

The problem state variable will be shown in Table 4.1 with columns 1 and 2 representing the various years (stages) and their corresponding state (age) variable(s) respectively.

Table 4.1: State variables for MMT bus

k	i
1	0,2
2	1,3
3	1,2,4
4	1,2,3,5
5	1,2,3,4,6
6	1,2,3,4,5,7
7	1,2,3,4,5,6,8
8	1,2,3,4,5,6,7,9
9	1,2,3,4,5,6,8,10
10	1,2,3,4,5,6,7,8,9,11

Since our goal is to maximize the total net profit, the MMT operational cost recursive relation is:

$$V_k(i) = max \begin{cases} I_k(i) - C_k(i) + V_{k+1}(i + 1) & Keep \\ I_k(0) - C_k(0) + R_k(i) + V_{k+1}(1) & Replace \end{cases}$$

With the recurrence relation in place, the final step of the solution procedure consists of the recursive computation of the value function $V_k(i)'s$ given the stages and states.

Microsoft Excel solver was used for solving the replacement problem as a dynamic programming model to find the optimal replacement policy which maximizes the net profit for the studied busses.

4.1.2 RESULTS

Table 4.2 below tabulates the optimal decision variable sequence for the studied buses as extracted from the last columns of the Microsoft Excel output of the tables in appendix A to E.

Table 4.2: Buses optimal decision variable sequence

Bus	1	2	3	4	5	6	7	8	9	10	11
Commuter	K	K	K	K	K	R	K	K	K	K	K
Neoplan 1	K	K	K	K	R	K	K	K	K	R	K
Neoplan 2	K	K	K	K	K	R	K	K	K	K	K
Daf	K	K	K	K	K	R	K	K	K	K	K
Jonckheere	K	K	K	K	K	R	K	K	K	K	K

K=Keep
R=Replace

This means that, the VDL Neoplan City (1^{st} Generation) bus comes with the optimal policy {K, K, K, K, R, K, K, K, K, R, K} with a corresponding total net profit of GH₵447780.00. MMT should keep the bus for the first four years of service and replaced at the beginning of the fifth year, then keep it till the start of the tenth year where it must be replaced again. It then follows with keep decisions till the end of the planned horizon.

The VDL Commuter, VDL Neoplan City (2nd Generation), VDL Daf and Jonckheere buses are characterized by keep actions in the first five years then followed by replace decisions at the start of year 6 with keep actions spanning to the end of the period as in optimal policy {K, K, K, K, K, R, K, K, K, K, K} thereby yielding GH₵303,845.00, GH₵419,900.00, GH₵271,733.00 and GH₵331,172.00 as optimal net profit throughout the planned horizon respectively.

Table 4.3 illustrates the profit/loss (in GH₵) associated with replace and the keep actions of each bus at the policy year. There are huge differences between the replace and keep values. Replacement carried out at the policy year can allow MMT to earn about 190% pure profit more than the keep options.

Table 4.3: Replace and Keep action profits/loss at given action years

Bus	Loss obtained from Keep	Profit obtained from Replace	Policy Year
Commuter	-4153	71486	6
Neoplan 1	-29138	35271	5
Neoplan 2	-15610	21550	6
Daf	-39712	17861	6
Jonckheere	-15926	19456	6

Fig 4.1 illustrates the comparison between replace and keep profit/loss at their respective policy years where the keep actions are characterised by negative values signifying loss to MMT should a replace decision be compromised at the policy year.

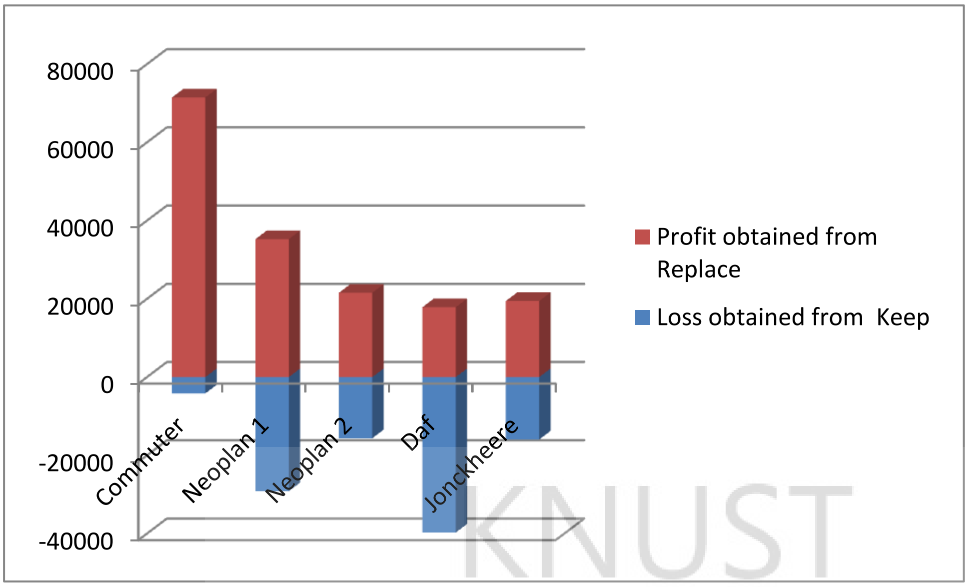


Figure 4.1: Replace and keep profit/loss comparison

4.2 DISCUSSION

Clearly, non adherence to the policy year replace action given the available data spells out the danger to MMT Ltd running at a loss. Keeping the VDL Jonckheere, VDL Daf, VDL Neoplan City (2^{nd} generation) and VDL Commuter buses without replacing them at the start of the sixth year of the planned horizon results in the following losses: GH₵15,926.00, GH₵39,712.00, GH₵15,610.00 and GH₵4,153.00 respectively. The VDL Neoplan (1^{st} Generation) however, registers a net loss of GH₵29,138.00 if MMT fails to replace it at the commencement of year 5 in its service life. Table 4.3 and figure 4.1 clearly through more light on this. It is however interesting to note that, adherence to the policy year replace action yielded not only the desired profit but also made it possible to unearth the individual bus's contribution to MMTs' total net profit thereby buttressing any such decision to endorse the usage of one kind of a bus over the other. On the other hand, the net profit realised should MMT stick to the policy year replace decision of replacing the VDL Jonckheere, VDL Daf, VDL Neoplan City (2^{nd} generation) and VDL Commuter buses is GH₵19,456.00, GH₵17,861.00, GH₵21,550.00 and GH₵71,486.00 respectively at the start of year 6 with the

VDL Neoplan City (1st Generation) seeing a net profit of GH₵35,271.00 at the start of year 5 of its policy year. On the other hand, the negative signs associated with the total net profits of non adherence to the optimal policy as in the case of the Commuter and Daf buses indicate loss to the company.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

The problem was modeled as a recursive function as shown in slide fourteen.

All buses should be replaced at the start of the sixth year as their policy year except Neoplan City (1st generation) bus which should be replaced at the start of the fifth year of the planned horizon.

Optimal replacement policies allows MMT Ltd (Kumasi depot) to earn GH₵165,624.00 in profit with the keep actions yielding a loss of GH₵104,539.00 It is noted that the optimal replacement policies allows MMT Ltd (Kumasi depot) to earn about 190% more than the keep actions according to the collected data.

5.2 RECOMMENDATION

Further research work using other methods is highly recommended to overcome the weakness in information, data and the predicted values to achieve more accurate policies.

It is recommended that the MMT Limited keep their books well for easy access to information and data.

Again, it is strongly recommended that MMT should dispose off all its buses stated herein after five (5) years of usage except the VDL Neoplan (1st Generation) which should be disposed off after four (4) of usage.

REFERENCES

Adda, Jerome, Cooper, Russel (2003) An accessible introduction to dynamic programming in economics, MIT Press, Cambridge.

Alexander, S. (1998) Theory of Linear Programming and Integer Programming. John Wiley and Sons, New York.

Amponsah, S.K. (2006) Optimization Techniques I, University Printing Press, KNUST, Kumasi.

Ayres, R.M. and J. Waizeneker (1978) A Practical Approach to Vehicle Replacement. *Chartered Mechanical Engineer*, Vol. 25, No. 9, pp. 73-75.

B. Bhowmik, "Design and Analysis of Algorithms – WBUT Series", 1st Edition, 2010 [in press].

Beaumont, N. (2007) Using dynamic programming to determine an optimal strategy in a contract bridge tournament. Journal of operational research society advance online publication 22/04/09.

Bellman, (1957) Dynamic programming, Princeton university press, New Jersey.

Blackwell, D. (1965) Positive Dynamic Programming. Proceedings of the 5th Berkeley Symposium 3 415.428.

Buddhakulsomsiri, J. and P. Parthanadee (2006) Parallel Replacement Problem for a Fleet with Dependent Use. Presented at the Industrial Engineering Research Conference (IERC), Orlando, FL.

Bush W. R., Pincus D. J. and Sielaff D. J. (1975) A static analyzer for finding dynamic programming errors, journal willey interscience, volume 30, issue 7, pp 775-802.

Chee, P.C.F. A Practical Vehicle Replacement Policy for Ontario Hydro. *Ontario Hydro Research Quarterly,* Vol. 27, No. 3, Third Quarter, pp. 3-6.

Chow, C.S. and Tsitsiklis, J.N. (1989) .The Complexity of Dynamic Programming. Journal of Complexity 5 466.488.

Daniel, P. W., Ellis (2007), Beat Tracking by Dynamic Programming, Journal of new music research, volume, 36, issue 1, pp51-60.

Davenport, N.S., M.D. Anderson, and P.A. Farrington (2005) Development and Application of a Vehicle Procurement Model for Rural Fleet Asset Management. *Transportation Research Record,* No. 1927, pp. 123-127.

David B. Wagner (1995) "Dynamic Programming", THE MATHEMATICA JOURNAL, Miller Freeman Publications.

David, R. A., Sweeney J. D., Thomas A. W. (1998) An introduction to management science, West Publishing Company, San Francisco.
Drinkwater, R.W. and N.A.J. Hastings (1967) An Economic Replacement Model. *Operational Research Society*, OR, Vol. 18, No. 2, pp. 121-138.

Eilon, S., J. R. King, and D. E Hutchinson (1966) A Study in Equipment Replacement; *Operational Research Society*, OR, Vol. 17, No. 1, pp. 59-71.

Ellis Horowitz, Sartaj Sahani, Sanguthevar Rajasekaran (2008) Computer Algorithms, University Press, 2nd Edn,.

Éva Tardos and Jon Kleinberg (2005) Algorithm Design, Addison Wesley.

Gupta A., Lawsirirat C. (2006), Strategically optimum maintenance of monitoring-enabled multi component systems using continuous-time jump deterioration model; Journal of Quality in Maintenance Engineering 12; 306-329.

Hartman, J. C. (1999) A General Procedure for Incorporating Asset Utilization Decisions into Replacement Analysis. *Engineering Economist*, Vol. 44, No. 3, pp. 217.

Hartman, J. C. (2004). Multiple Asset Replacement Analysis Under Variable Utilization and Stochastic Demand. *European Journal of Operational Research*, Vol. 159, No. 1, pp. 145-165.

Hastings, N.A.J. (1969) The Repair Limit Replacement Method. *Operational Research Society*, OR, Vol. 20, No. 3, pp. 337-349.

Herbert S. Wilf (1994). Algorithms and Complexity, University of Pennsylvania, Internet Edition, Summer,

Hillier F. S. and Gerald J.L. (2005). Introduction to operations research. McGraw-Hill, New York.

Hotelling, H., (1925). A general mathematical theory of depreciation, Journal of the American Statistical Association, McGraw-Hill book Company, New York.

J. N. Tsitsiklis and B. Van Roy (1999). "Optimal Stopping of Markov Processes: Hilbert Space Theory, Approximation Algorithms, and an Application to Pricing Financial Derivatives", IEEE Transactions on Automatic Control; Vol. 44, No. 10, pp. 18401851.

Jardine A.K.S., Banjevic D., Makis V. (1997), Optimal replacement policy and the structure of software for condition-based maintenance; *Journal of Quality in Maintenance Engineering* 3; 109-119.

John Rust (2006). Dynamic Programming, University of Maryland.

John W. Chinneck(2006) Practical Optimization: a Gentle Introduction, http://www.sce.carleton.ca/faculty/chinneck/po.html , 23/11/2010, 12:00 pm

Khasnabis, S., J. Bartus, and R.D. Ellis (2003) Asset Management Framework for State Departments of Transportation to Meet Transit Fleet Requirements. *Transportation Research Record*, No. 1835, pp. 74-83.

Love, C.E., R. Rodger, and G. Blazenko (1982). Repair Limit Policies for Vehicle Replacement. *INFOR Journal,* Vol. 20, pp. 226-37.

M.H. Alsuwaiyel (2002). “Algorithms Design Techniques and Analysis”, PHEI.

Mahmut, P. (2000), Interactive Operations Research with Maple: Methods and Models. Bikhauser, Boston.

Nakagawa, T. and S. Osaki (1974). The Optimum Repair Limit Replacement Policies. *Operational Research Quarterly*, Vol. 25, No. 2, pp. 311-317.

Nicole, E. K. and Marie-Claire Quenez (1995), Dynamic programming of contingent claims incomplete market: SIAM J control and optimization volume 33, issue 1, pp 29-66.

Normal J M and Clarke S. R. (2004), dynamic programming in cricket: optimizing batting order for sticky wicket, journal of the Operational Research Society, volume 48, pp 1678-1682.

Oren Weimann (2009) Accelerating Dynamic Programming, Department of Electrical Engineering and Computer Science.

Preinreich, G. D., (1940) The economic life of industrial equipment, econometrical.

Professor Bergin (1998). “Lecture Notes on Dynamic Programming”, Economics 200E, Spring.

Quansong R; Steele A. J., Schwalbach M. S., Fuhrman J. A and Fengzhu S. (2006) A dynamic programming for binning microbial community profiles. Journal on dynamic programming in Bioinformatics, volume 22, pp 1508-1514.

Redmer, A. (2005) Vehicle Replacement Planning in Freight Transportation Companies. Proceedings of the 16th Mini - EURO Conference and 10th Meeting of EURO Working Group Transportation, Poznan, Poland.

Rees, L.P., E.R. Clayton, and B.W. III Taylor (1982) Network Simulation Model for Police Patrol Vehicle Maintenance and Replacement Analysis. *Comps. Environ. & Urban Sys*, Vol. 7, No. 3, pp. 191-196l.

Robert Sedgewick(1984) Algorithms, ADDISON-WESLEY PUBLISHING COMPANY.

Ronald A. Howard (1960) Dynamic Programming and Markov Processes, John Wiley & Sons,.

Ross, Sheldon M. (1983) Introduction to Stochastic Dynamic Programming. San Diego: Academic Press.

Sherwin J.D. (1999) Age-based opportunity maintenance; Journal of Quality in Maintenance Engineering 5; 221-235.

Sherwin J.D., Al-Najjar B. (1999) Practical models for condition-based monitoring inspection intervals; Journal of Quality in Maintenance Engineering 5; 203-209.

Simms, B.W., B.G. Lamarre, and A.K.K. Jardine (1984) Optimal Buy, Operate and Sell Policies for Fleets of Vehicles. *European Journal of Operational Research*, Vol. 15, No. 2, pp. 183-195.

Sinuany-Stern Z.S. (1993), Replacement policy under partially observed Markov process; International Journal of Production Economics 29; 159-166.

Sinuany-Stern Z.S., David I., Biran S. (1997), An efficient heuristic for a partially observable Markov decision process of machine replacement; Journal of Computers in Operations Research 24; 117-126.

Slater L. J. (1964) A dynamic programming process, journal on dynamic programming in the computer, volume 7, pp 36-39.

Smith K. D, Pass M, (1997) Dynamic programming process: an introduction, MIT press, Cambridge.

Steven Skiena (2007) Applications of Dynamic Programming, State University of New York Stony Brook, NY 11794–4400.pp 13-20.

Streufert, P. (1998) Recursive Utility and Dynamic Programming. in S. Barbera, P. Hammond, and C. Seidl (editors) Handbook of Utility Theory, Volume 1 chapter 3 93.121.

Taylor, J. S., (1923) A statistical theory of depreciation, Journal of the American Statistical Association, McGraw-Hill book Company, New York.

Thomas H. Coremen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein(2008) Introduction to Algorithms, PHI, 2nd Edn,.

Traub, J.F. and Werschulz (1998) Complexity and Information Cambridge University Press, Cambridge, UK.

Vijaya Ramachandran (2006) ALGORITHMS, The University of Texas, Austin.

Wang C.H., Hwang S.L. (2004), A stochastic maintenance management model with recovery factor;Journal of Quality in Maintenance Engineering 10; 154-164.

Weissmann, J., A.J. Weissmann, and S. Gona (2003) Computerized Equipment Replacement Methodology. *Transportation Research Record*, No. 1824, pp. 77-83.

Zeqing L. and Shin M. K. (2006) Properties of solutions for certain functional equations arising in dynamic programming, journal of global optimization, volume 34, issue 2, pp 273-292.

Zhou X., Xi L., Lee J. (2006), A dynamic opportunistic maintenance policy for continuously monitored systems; Journal of Quality in Maintenance Engineering 12; 294-305.

DATA OBTAINED FROM MMT LIMITED

Income Matrix of Buses in Ghana Cedis

BUS/YEAR	2006	2007	2008	2009	2010
COMMUTER	98073	97824	84897	85021	84524
NEOPLAN 1	90000	78000	70800	71400	73000
NEOPLAN 2	78900	78700	68300	68400	68000
DAF	97662	95940	85854	83517	79704
JONCHKEERE	98625	98375	85375	85500	85000

Cost Matrix of Buses in Ghana Cedis

BUS/YEAR	2006	2007	2008	2009	2010
COMMUTER	19944	21163	28697	32132	32021
NEOPLAN 1	12000	17000	24000	22100	27600
NEOPLAN 2	18000	19100	25900	29000	28900
DAF	23370	24108	36654	30873	38253
JONCHKEERE	21654	22977	31158	34887	34767

Replacement Cost Matrix of Buses in Ghana Cedis

BUS/YEAR	2006	2007	2008	2009	2010
COMMUTER	177840	179400	195000	195000	199680
NEOPLAN 1	110000	115000	125000	125000	128000
NEOPLAN 2	114000	115000	125000	125000	128000
DAF	189240	189750	190000	191250	193280
JONCHKEERE	189240	189750	190000	191250	193280

APPENDIX B

Ms Excel output for VDL Neoplan city 1st generation bus

Stage 11 2016

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
12	42050	57060	141050	-15010	-157150	-15010	Keep
10	42700	54800	138000	-12100	-85000	-12100	Keep
9	49500	50400	138000	-900	-85000	-900	Keep
8	58000	49900	138000	8100	-85000	8100	Keep
7	74400	48400	138000	26000	-85000	26000	Keep
6	74600	46100	138000	28500	-85000	28500	Keep
5	65000	36500	138000	28500	-85000	28500	Keep
4	75900	34600	130000	41300	-77000	41300	Keep
3	86700	22800	127000	63900	-74000	63900	Keep
2	86800	20000	126000	66800	-73000	66800	Keep
1	86900	18700	125000	68200	-72000	68200	Keep
0	87000	17900					

Stage 10 2015

i	$I_k(i)$	$C_k(i)$	$R_k(i)$		Vk	Vr	$V_k(i)$
11	42700	54800	138000	-27110	40313	40313	Replace
9	49500	50400	138000	-13000	40313	40313	Replace
8	58000	49900	138000	7200	40313	40313	Replace
7	74400	48400	138000	34100	40313	40313	Replace
6	74600	46100	138000	-54500	40313	40313	Replace
5	65000	36500	138000	40313	57000	57000	Replace
4	75900	34600	130000	69800	48313	69800	Keep
3	86700	22800	127000	105200	51313	105200	Keep
2	86800	20000	126000	130700	52313	130700	Keep
1	86900	18700	125000	135000	53313	135000	Keep
0	87000	17900					

Stage 9 2014

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
10	25700	36800	135000	29213	42316	42316	Replace
8	36600	35900	135000	41013	42316	42316	Replace
7	37800	35500	135000	42613	42316	42613	Keep
6	37900	33800	135000	44413	42316	44413	Keep
5	52000	31300	135000	75200	42316	75200	Keep
4	55800	28300	126000	84500	51316	84500	Keep
3	69300	24200	125000	114900	52316	114900	Keep
2	72600	20200	124000	157600	53316	157600	Keep
1	78600	17500	124000	191800	53316	191800	Keep
0	81700	16800					

Stage 8 2013

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
9	29800	37600	131000	34516	34749	34749	Replace
7	35200	35400	131000	42116	34749	42116	Keep
6	39600	34100	131000	48113.2	34749	48113	Keep
5	41400	34000	131000	51813.2	34749	51813	Keep
4	51800	31900	125000	95100	40749	95100	Keep
3	59400	25500	124000	118400	41749	118400	Keep
2	68500	24200	124000	159200	41749	159200	Keep
1	78200	21700	123000	214100	42749	214100	Keep
0	79900	21600					

Stage 7 2012

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
8	37900	39700	131000	-32948	255080	255080	Replace
6	40900	36700	131000	-46316	255080	255080	Replace
5	50400	36500	131000	-62013	255080	255080	Replace
4	54700	33200	131000	-73313	255080	255080	Replace
3	60500	32700	125000	-12290	261080	261080	Replace

2	69300	21700	124000	16600	-262080	16600	Keep
1	72700	18600	124000	21330	-262080	21330	Keep
0	80400	18100					

Stage 6 2011

i	$I_k(i)$	$C_k(i)$		$R_k(i)$	Vk	Vr	$V_k(i)$
7	37300	40800	128000	251580	25100	251580	Keep
5	43200	38000	128000	260280	25100	260280	Keep
4	56800	37900	123000	273980	30100	273980	Keep
3	59100	34600	123000	279580	30100	279580	Keep
2	63900	34200	122000	290780	31100	290780	Keep
1	76200	22100	122000	316180	31100	316180	Keep
0	81700	19500					

Stage 5 2010

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
6	68100	28300	128000	-29138	35271	35271	Replace
4	73000	27600	126000	-30568	35471	35471	Replace
3	73300	26600	125000	-32068	35571	35571	Replace
2	77100	26300	122000	-33038	35871	35871	Replace
1	80800	19600	121000	-35198	35971	35971	Replace
0	82200	18200					

Stage 4 2009

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
5	68000	28500	125000	291080	173219	291080	Keep
3	71400	22100	125000	309580	173219	309580	Keep
2	72700	21800	122000	324880	176219	324880	Keep
1	77600	21500	121000	335680	177219	335680	Keep
0	79600	18100					

Stage 3 2008

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
4	67400	38000	125000	320480	269580	320480	Keep
2	70800	24000	122000	356380	272580	356380	Keep
1	77000	19000	120000	382880	274580	382880	Keep
0	77500	18600					

Stage 2 2007

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
3	72300	19000	115000	373780	-115000	373780	Keep
1	78000	17000	110000	417380	-110000	417380	Keep
0	77000	11000					

Stage 1 2006

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
2	89000	15000	110000	447780	229380	447780	Keep
0	90000	12000					

APPENDIX C

Ms Excel output for VDL Neoplan city 2nd generation bus

Stage 11 2016

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
12	44250	80450	142000	-36200	-38000	-36200	Keep
10	45000	78800	138000	-33800	-34000	-33800	Keep
9	57000	78300	138000	-21300	-34000	-21300	Keep
8	63000	53000	138000	10000	-34000	10000	Keep
7	65000	46000	138000	19000	-34000	19000	Keep
6	60500	42500	138000	18000	-34000	18000	Keep
5	78000	35500	138000	42500	-34000	42500	Keep
4	74000	22100	130000	51900	-26000	51900	Keep
3	72500	22000	127000	50500	-23000	50500	Keep
2	80000	21900	126000	58100	-22000	58100	Keep
1	80000	21800	125000	58200	-21000	58200	Keep
0	80500	21500					

Stage 10 2015

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
11	45000	78800	138000	-70000	69800	69800	Replace
9	57000	78300	138000	55100	-69800	55100	Keep
8	63000	53000	138000	11300	-69800	11300	Keep
7	65000	46000	138000	29000	-69800	29000	Keep
6	60500	42500	138000	37000	-69800	37000	Keep
5	78000	35500	138000	60500	-69800	60500	Keep
4	74000	22100	130000	94400	-61800	94400	Keep
3	72500	22000	127000	102400	-58800	102400	Keep
2	80000	21900	126000	108600	-57800	108600	Keep
1	80000	21800	125000	116300	-56800	116300	Keep
0	80500	21500					

Stage 9 2014

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
10	36500	46000	138000	-79300	48900	48900	Replace
8	47000	36300	138000	-44400	48900	48900	Replace
7	58000	32000	138000	-14700	48900	48900	Replace
6	68500	27400	138000	70100	48900	70100	Keep
5	73500	24000	138000	86500	48900	86500	Keep
4	74300	23000	130000	111800	56900	111800	Keep
3	78500	18500	125000	154400	61900	154400	Keep
2	80000	17000	124000	165400	62900	165400	Keep
1	82500	15500	124000	175600	62900	175600	Keep
0	85500	14900					

Stage 8 2013

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
9	33500	37000	131000	45400	111100	111100	Replace
7	37000	36000	131000	49900	111100	111100	Replace
6	48500	27000	131000	70400	111100	111100	Replace
5	55500	26700	131000	98900	111100	111100	Replace
4	69000	26400	125000	129100	117100	129100	Keep
3	70500	24000	124000	158300	118100	158300	Keep
2	76100	22000	124000	208500	118100	208500	Keep
1	78200	21000	123000	222600	119100	222600	Keep
0	85500	19000					

Stage 7 2012

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
8	27000	29000	130000	-10910	161500	161500	Replace
6	38000	27000	130000	-12210	161500	161500	Replace
5	61000	26500	130000	-14560	161500	161500	Replace
4	72000	25000	128000	-15810	163500	163500	Replace
3	73000	24000	125000	178100	166500	178100	Keep
2	78000	21000	123000	215300	168500	215300	Keep
1	85000	18000	122000	275500	169500	275500	Keep

0	86000	17100					

Stage 6 2011

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
7	35600	41000	128000	**-15610**	**215500**	**215500**	**Replace**
5	40800	37000	128000	-16530	215500	215500	Replace
4	58000	35000	123000	-18450	220500	220500	Replace
3	63000	29600	123000	-19690	220500	220500	Replace
2	78000	24000	122000	232100	221500	232100	Keep
1	80500	21000	122000	274800	221500	274800	Keep
0	86000	18000					

Stage 5 2010

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
6	56400	31500	128000	240400	208200	240400	Keep
4	68000	28900	126000	254600	210200	254600	Keep
3	73000	24800	125000	268700	211200	268700	Keep
2	75400	20900	122000	275000	214200	275000	Keep
1	76900	19300	121000	289700	215200	289700	Keep
0	79800	18400					

Stage 4 2009

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
5	67100	32000	125000	275500	173219	275500	Keep
3	68400	29000	125000	294000	173219	294000	Keep
2	69800	25000	122000	313500	176219	313500	Keep
1	74500	19000	121000	330500	177219	330500	Keep
0	79200	18500					

Stage 3 2008

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
4	65700	30700	125000	310500	151780	310500	Keep
2	68300	25900	122000	336400	154780	336400	Keep

1	77500	19300	120000	371700	156780	371700	Keep
0	79200	18000					

Stage 2 2007

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
3	69500	19500	115000	360500	201880	360500	Keep
1	78700	19100	110000	396000	206880	396000	Keep
0	79000	18300					

Stage 1 2006

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
2	78600	19200	114000	419900	225380	419900	Keep
0	78900	18000					

APPENDIX D

Ms Excel output for **VDL Jonckheere bus**

2016

i	$I_k(i)$	$C_k(i)$	$R_k(i)$	Vk	Vr	$V_k(i)$	D_k
12	50328	152740	223463	10241	-242563	10241	KeeP
10	56250	94796	212520	38546	-23162	38546	KeeP
9	71250	94195	212520	-22945	-23162	-22945	KeeP
8	78750	63759	212520	14991	-23162	14991	KeeP
7	81250	55338	212520	25912	-231620	25912	KeeP
6	75625	51128	212520	24498	-231620	24498	KeeP
5	97500	42707	212520	54794	-231620	54794	KeeP
4	92500	26586	200200	65914	-219300	65914	KeeP
3	90625	26466	195580	64159	-214680	64159	KeeP
2	100000	26346	194040	73654	-213140	73654	KeeP
1	100000	26225	192500	73775	-211600	73775	KeeP
0	100625	25865					

11	56250	94796	212520	140958	-144320	140958	KeeP
9	71250	94195	212520	61491	-144320	61491	KeeP
8	78750	63759	212520	7954	-144320	7954	KeeP
7	81250	55338	212520	40903	-144320	40903	KeeP
6	75625	51128	212520	50410	-144320	50410	KeeP
5	97500	42707	212520	79291	-144320	79291	KeeP
4	92500	26586	200200	120707	-132000	120707	KeeP
3	90625	26466	195580	130073	-127380	130073	KeeP
2	100000	26346	194040	137813	-125840	137813	KeeP
1	100000	26225	192500	147429	-124300	147429	KeeP
0	100625	25865	0				

10	45625	55338	201480	150671	-201480	150671	KeeP

8	58750	49669	201480	52410	-201480	52410	KeeP
7	72500	48496	201480	16050	-201480	16050	KeeP
6	85625	32962	201480	93566	-201480	93566	KeeP
5	91875	28872	201480	113413	-201480	113413	KeeP
4	92875	27669	189800	144497	-189800	144497	KeeP
3	98125	22256	182500	196577	-182500	196577	KeeP
2	100000	20451	181040	209622	-181040	209622	KeeP
1	103125	18647	181040	222292	-181040	222292	KeeP
0	106875	17925					

9	41875	44511	199120	153307	-199120	153307	KeeP
7	46250	43308	199120	49468	-199120	49468	KeeP
6	60625	32481	199120	44194	-199120	44194	KeeP
5	69375	32120	199120	130821	-199120	130821	KeeP
4	86250	31759	190000	167903	-190000	167903	KeeP
3	88125	28872	188480	203750	-188480	203750	KeeP
2	95125	26466	188480	265236	-188480	265236	KeeP
1	97750	25263	186960	282109	-186960	282109	KeeP
0	106875	22857					

8	33750	34887	198900	154444	-198900	154444	KeeP
6	47500	32481	198900	34449	-198900	34449	KeeP
5	76250	31880	198900	88565	-198900	88565	KeeP
4	90000	30075	195840	190746	-195840	190746	KeeP
3	91250	28872	191250	230281	-191250	230281	KeeP
2	97500	25263	188190	275987	-188190	275987	KeeP
1	106250	21654	186660	349832	-186660	349832	KeeP
0	107500	20571	0				

7	44500	49323	194560	-15926	19456	19456	Replace
5	51000	44511	194560	-27960	19456	19456	Replace
4	72500	42105	186960	11896	-18696	11896	KeeP
3	78750	35609	186960	23388	-18696	23388	KeeP
2	97500	28872	185440	29890	-18544	29890	KeeP
1	100625	25263	185440	35134	-18544	35134	KeeP
0	107500	21654	0				

6	70500	37895	193280	126662	-193280	126662	KeeP
4	85000	34767	190260	22273	-190260	22273	KeeP
3	91250	29834	188750	180375	-188750	180375	KeeP
2	94250	25143	184220	302994	-184220	302994	KeeP
1	96125	23218	182710	371816	-182710	371816	KeeP
0	99750	22135					

5	83875	38496	191250	81283	-191250	81283	KeeP
3	85500	34887	191250	72886	-191250	72886	KeeP
2	87250	30075	186660	237550	-186660	237550	KeeP
1	93125	22857	185130	373262	-185130	373262	KeeP
0	99000	22256					

4	82125	36932	190000	36090	-190000	36090	KeeP
2	85375	31158	185440	127103	-185440	127103	KeeP
1	96875	23218	182400	311207	-182400	311207	KeeP
0	99000	21654					

Stage 2

3	86875	23459	189750	27327	-189750	27327	KeeP
1	98375	22977	181500	202501	-181500	202501	KeeP
0	98750	22015					

Stage 1

2	98250	23098	189240	102479	-189240	102479	KeeP
0	98625	21654					

APPENDIX E
Ms Excel output for VDL commuter bus

2016							
12	54520	61590	225000	-7071	-297240	-7071	Keep
10	55935	87310	225000	-31375	-297240	-31375	Keep
9	70851	86756	225000	-15905	-297240	-15905	Keep
8	78309	58724	215280	19585	-287520	19585	Keep
7	80795	50968	215280	29827	-287520	29827	Keep
6	75202	47090	215280	28112	-287520	28112	Keep
5	96954	39334	215280	57620	-287520	57620	Keep
4	91982	24487	202800	67495	-275040	67495	Keep
3	90118	24376	198120	65742	-270360	65742	Keep
2	99440	24265	196560	75175	-268800	75175	Keep
1	99440	24154	195000	75286	-267240	75286	Keep
0	100062	23822					
	Stage 10	2015					
11	55935	87310	215280	-38446	-216234	-38446	Keep
9	70851	86756	215280	-47281	-216234	-47281	Keep
8	78309	58724	215280	3680	-216234	3680	Keep
7	80795	50968	215280	49412	-216234	49412	Keep
6	75202	47090	215280	57939	-216234	57939	Keep
5	96954	39334	215280	85732	-216234	85732	Keep
4	91982	24487	202800	125115	-203754	125115	Keep
3	90118	24376	198120	133237	-199074	133237	Keep
2	99440	24265	196560	140917	-197514	140917	Keep
1	99440	24154	195000	150460	-195954	150460	Keep
0	100062	23822					
10	45370	50968	215280	-44045	-154587	-44045	Keep
8	58421	40220	215280	-29080	-154587	-29080	Keep
7	72094	35456	215280	40318	-154587	40318	Keep
6	85146	30359	215280	104198	-154587	104198	Keep

5	91361	26592	215280	122708	-154587	122708	Keep
4	92355	25484	202800	152603	-142107	152603	Keep
3	97576	20498	195000	202193	-134307	202193	Keep
2	99440	18836	193440	213841	-132747	213841	Keep
1	102548	17174	193440	226290	-132747	226290	Keep
0	106277	16509					

9	41641	40996	204360	-43400	-63294	-43400	Keep
7	45991	39888	204360	-22977	-63294	-22977	Keep
6	60286	29916	204360	70687	-63294	70687	Keep
5	68987	29584	204360	143601	-63294	143601	Keep
4	85767	29251	195000	179223	-53934	179223	Keep
3	87632	26592	193440	213642	-52374	213642	Keep
2	94592	24376	193440	272409	-52374	272409	Keep
1	97203	23268	191880	287776	-50814	287776	Keep
0	106277	21052					

8	33561	32132	202800	-41971	-2975	-2975	Replace
6	47234	29916	202800	-5659	-2975	-2975	Replace
5	75823	29362	202800	117148	-2975	117148	Keep
4	89496	27700	199680	205397	145	205397	Keep
3	90739	26592	195000	243370	4825	243370	Keep
2	96954	23268	191880	287328	7945	287328	Keep
1	105655	19944	190320	358120	9505	358120	Keep
0	106898	18947					

7	44251	45428	199680	-4153	71486	71486	Replace
5	50714	40996	199680	6743	71486	71486	Replace
4	72094	38780	191880	150462	79286	150462	Keep
3	78309	32797	191880	250909	79286	250909	Keep
2	96954	26592	190320	313732	80846	313732	Keep
1	100062	23268	190320	364122	80846	364122	Keep
0	106898	19944					

6	70105	34902	199680	106689	85638	106689	Keep
4	84524	32021	196560	-123989	167562	167562	Replace
3	90739	27478	195000	213723	169122	213723	Keep
2	93722	23157	190320	321474	173802	321474	Keep
1	95587	21384	188760	387935	175362	387935	Keep
0	99191	20387					
5	83405	35456	195000	154639	114987	154639	Keep
3	85021	32132	195000	220451	114987	220451	Keep
2	86761	27700	190320	272784	119667	272784	Keep
1	92604	21052	188760	393026	121227	393026	Keep
0	98446	20498					
stage	**3**						
4	81665	34016	195000	162637	119524	162637	Keep
2	84897	28697	190320	171187	124204	171187	Keep
1	96333	21384	187200	194615	127324	194615	Keep
0	98446	19944					
stage	**2**						
3	86389	21606	179400	227419	-62706	227419	Keep
1	97824	21163	171600	247848	-54906	247848	Keep
0	98197	20276					
stage	**1**						
2	97700	21274	177840	303845	-8121	303845	Keep
0	98073	19944					

APPENDIX F
Ms Excel output for VDL Daf bus

12	62950	93500	235600	-30550	-320537	-30550	keep
10	65928	66912	212520	-984	-297457	-984	keep
9	62238	63468	212520	-1230	-297457	-1230	keep
8	97293	61377	212520	35916	-297457	35916	keep
7	90282	59901	212520	30381	-297457	30381	keep
6	86469	53628	212520	32841	-297457	32841	keep
5	99015	48831	212520	50184	-297457	50184	keep
4	101229	46986	200200	54243	-285137	54243	keep
3	104673	38253	195580	66420	-280517	66420	keep
2	106641	20418	194040	86223	-278977	86223	keep
1	107010	19557	192500	87453	-277437	87453	keep
0	107379	18942					

	stage	**8**					
11	65928	66912	212520	-31534	-213504	-31534	keep
9	62238	63468	212520	-2214	-213504	-2214	keep
8	97293	61377	212520	34686	-213504	34686	keep
7	90282	59901	212520	66297	-213504	66297	keep
6	86469	53628	212520	63222	-213504	63222	keep
5	99015	48831	212520	83025	-213504	83025	keep
4	101229	46986	200200	104427	-201184	104427	keep
3	104673	38253	195580	120663	-196564	120663	keep
2	106641	20418	194040	152643	-195024	152643	keep
1	107010	19557	192500	173676	-193484	173676	keep
0	107379	18942					

	stage	7					
9	32718	46986	201480	-45802	-114888	-45802	keep
8	41451	44526	201480	-5289	-114888	-5289	keep
7	44772	38868	201480	40590	-114888	40590	keep
6	71094	36285	201480	101106	-114888	101106	keep
5	76383	29643	201480	109962	-114888	109962	keep

4	81303	29397	189800	134931	-103208	134931	keep
3	85854	24477	182500	165804	-95908	165804	keep
2	92619	23247	181040	190035	-94448	190035	keep
1	100860	17589	181040	235914	-94448	235914	keep
0	104673	17589					

	stage	**8**					
9	35916	46002	199120	-55888	-37867	-37867	replace
7	48954	42312	199120	1353	-37867	1353	keep
6	49200	36408	199120	53382	-37867	53382	keep
5	55350	33702	199120	122754	-37867	122754	keep
4	58917	29274	190000	139605	-28747	139605	keep
3	60639	26445	188480	169125	-27227	169125	keep
2	67281	26199	188480	206886	-27227	206886	keep
1	75030	24231	186960	240834	-25707	240834	keep
0	98031	23370					

	stage	**7**					
8	53259	45633	198900	-30241	-33711	-30241	keep
6	58548	37392	198900	22509	-33711	22509	keep
5	61377	37269	198900	77490	-33711	77490	keep
4	69003	29520	195840	162237	-30651	162237	keep
3	76137	26445	191250	189297	-26061	189297	keep
2	80565	24969	188190	224721	-23001	224721	keep
1	96555	23739	186660	279702	-21471	279702	keep
0	98523	22878					

	stage	**6**					
7	41328	50799	194560	-39712	17861	17861	replace
5	41451	49077	194560	14883	17861	17861	replace
4	51783	47232	186960	82041	25461	82041	keep
3	60147	42558	186960	179826	25461	179826	keep
2	66420	36654	185440	219063	26981	219063	keep
1	79335	34563	185440	269493	26981	269493	keep
0	90774	23493					

	stage	**5**					
6	70479	38499	193280	49841	7087	49841	keep
4	79704	38253	190260	59312	10107	59312	keep
3	79827	36408	188750	125460	11617	125460	keep
2	79950	30258	184220	229518	16147	229518	keep
1	82779	22632	182710	279210	17657	279210	keep
0	91020	21894					
	stage	**4**					
5	80442	30996	191250	99287	10347	99287	keep
3	83517	30873	191250	111956	10347	111956	keep
2	84501	24600	186660	185361	14937	185361	keep
1	90651	23739	185130	296430	16467	296430	keep
0	100368	22755					
	stage	**3**					
4	71586	37761	190000	133112	31892	133112	keep
2	85854	36654	185440	161156	36452	161156	keep
1	96309	26814	182400	254856	39492	254856	keep
0	98892	24354					
	stage	**2**					
3	92619	27060	189750	198671	-8694	198671	keep
1	95940	24108	181500	232988	-444	232988	keep
0	97170	23370					
	stage	**1**					
2	97416	24354	189240	271733	-30544	271733	keep
0	97662	23370					