FEEDER ALLOCATION POLICY FOR A TURRET HEAD PLACEMENT MACHINE

USING DYNAMIC PROGRAMMING

Except where reference is made to the work of others, the work described in this thesis is my own or was done in collaboration with my advisory committee. This thesis does not include proprietary or classified information.

_____

Harish Krishna Ramaswamy

Certificate of Approval:

_____	_____
Dr. Jeffrey S. Smith Professor Industrial and Systems Engineering	Dr. Jorge Valenzuela, Chair Associate Professor Industrial and Systems Engineering
_____	_____
Dr. Emmett Lodree Assistant Professor Industrial and Systems Engineering	Stephen L. McFarland Acting Dean Graduate School

FEEDER ALLOCATION POLICY FOR A TURRET HEAD PLACEMENT MACHINE USING DYNAMIC PROGRAMMING

Harish Krishna Ramaswamy

A Thesis

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Master of Science

Auburn, Alabama
December 16, 2005

FEEDER ALLOCATION POLICY FOR A TURRET HEAD PLACEMENT MACHINE

USING DYNAMIC PROGRAMMING

Harish Krishna Ramaswamy

_____

Signature of Author

_____

Date of Graduation

VITA

Harish Krishna Ramaswamy, son of Ramaswamy Krishna and Bhama Narayanaswamy, was born April 1981, in Chennai, Tamilnadu, India. He graduated from Spartan Matriculation Higher Secondary School in 1999. He attended Sri Venkateswara College of Engineering, University of Madras for four years and graduated with a Bachelor of Engineering degree in Mechanical Engineering in May 2003. He then entered the Graduate School at Auburn University, Department of Industrial and Systems Engineering, in August 2003.

THESIS ABSTRACT

FEEDER ALLOCATION POLICY FOR A TURRET HEAD PLACEMENT MACHINE USING DYNAMIC PROGRAMMING

Harish Krishna Ramaswamy
Master of Science, December 16, 2005
(B.E, Madras University, 2003)

85 Typed Pages

Directed by Dr. Jorge Valenzuela

The first objective of this thesis is to determine a feeder carriage allocation policy for a turret head placement machine. The feeder allocation problem deals with determining which component types have to be allocated to the surplus slots of the feeder carriage of the placement machine, so as to minimize the number of replenishments. The feeder allocation problem has been formulated as a dynamic programming model which maximizes the number of boards that can be made by a single allocation of component reels to the feeder slots. It has also been shown that maximizing the number of boards that can be made by a single allocation of components, delays replenishment as much as possible, thereby decreasing the total number of replenishments required.

As a second objective, a methodology of solving optimization problems with spreadsheets is proposed. This technique is called spreadsheet based optimization. The methodology is illustrated by considering the component allocation problem for a turret head placement machine.

ACKNOWLEDGMENTS

I would like to take this opportunity to thank one and all who have made this thesis possible.

First and foremost I would like to express my sincere gratitude to Dr. Jorge Valenzuela. I am deeply indebted to him for all his stimulating suggestions, encouragement, and support rendered to me, during my academic program in Auburn. He has been a wealth of information and my inspiration.

I wish to express my sincere appreciation to my committee members Dr. Jeffrey Smith and Dr. Emmett Lodree for their input and suggestions.

I am most grateful to my parents, grandparents, sister and brother in law for their love and support they have given me throughout my life. I am also ever thankful to all my friends for supporting me and encouraging me. Special thanks are however due to Skylab Gupta for all his help and support.

Style manual or journal used: European Journal of Operational Research

Computer software used: Microsoft Word, Microsoft Excel, CPLEX 8

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In the recent past the electronics manufacturing industry has seen remarkable innovations. This can be attributed to the highly demanding and competitive market of the electronics industry. The use of Printed Circuit boards (PCB's) in various applications has increased drastically. For example PCB's are extensively used in computers, mobile phones, cars, and robots. In the last year alone PCB production rate has increased by around 30% leading to annual production amount of more than 35 billion dollars [21, 19]. Hence in order to survive in this highly competitive environment, companies must be able to produce products faster, cheaper, and of better quality. Companies using through hole technology (THT) to manufacture PCB's found it difficult to keep up with the ever increasing demand. The advent of surface mount technology has greatly helped companies to manufacture products with complicated designs with high speed and quality.

1.1 Surface Mount Technology

Previously PCB's were manufactured by through hole technology (THT). In THT, components are mounted on the PCB's by inserting the leads of the components into the

holes made on the PCB board and then soldering the leads on the bottom of the board. In SMT the components to be mounted on the PCB are placed directly on one side or both sides of the bare substrate by soldering. Thus in the SMT the solder paste serves a dual purpose of holding the components on the board and providing electrical connectivity. The advantages of SMT over THT are obvious; holes need not have to drilled, smaller components can be placed easily, higher board density, and components can be placed on either side of the board. All these lead to a better quality and reduced costs.

The basic SMT manufacturing line consists of three steps [22]. In the first step the solder paste is screen printed on the bare substrate and inspection is carried out to check for defects. In the second step, the components to be mounted are placed on the board such that the leads of the components rest on the solder pads. In the third step, by means of the solder reflow process, electrical and mechanical contact of the leads with the board is established.

The components on the PCB board are usually placed by means of highly automated placement machines.

1.2 Turret head placement machine

The processes involved in SMT are highly automated. The placement of components on the board is usually done by placement machines. Basically there are two types of placement machines [18]; pick and place (PAP) machine and the turret head placement machine also known as the chip shooter machine. Each of these machines has its own characteristics. In the PAP machine, multiple stationary feeders are used to store the components. A moving placement head is used to pick up the components from the

feeder slots one at a time and place it on the board which is also stationary. PAP machines operate at a lower speed than chip shooter machines, but have better accuracy and are usually used to place large components.

This thesis is based on the turret head placement machine. The turret head placement machine it characterized by its high placement speed; it can place between 40,000 and 53,000 components per hour [14]. A turret head placement machine is shown in Figure 1.1.

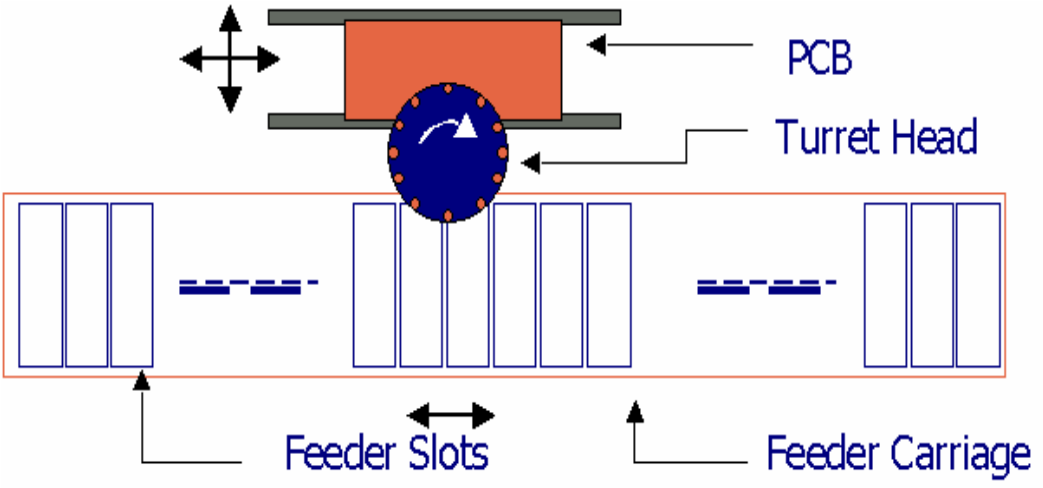

Figure 1.1: Turret head placement machine

A turret head placement machine consists of a rotating turret, a moving feeder carriage and movable X-Y table on which the PCB is placed. A reel of each component type is placed in each slot of the feeder carriage. Sometimes two or more slots may be required to hold the reels of a component. The rotating turret usually has around 10 to 12 heads [18] and picks up components from the feeder carriage on one side while simultaneously placing the components on the PCB. The assembly heads each have several nozzles of different sizes. The moving X-Y table facilitates the positioning of the

PCB. Turret head placement machines are usually used to place small components on the board.

1.3 Problem Overview

In the assembly of PCB's, the placement operations are usually the bottlenecks and are also the most time consuming task. Any reduction in the placement time would greatly reduce the cycle time and therefore improve productivity.

There are three sub problems involved in the optimization of placement problems [5]. The first problem deals with determination of the component placement sequence. The second problem involves assigning component types to the feeder slots, and the third problem deals with the sequence in which each components are to be retrieved.

This research deals with the second sub problem. In the turret head placement machine when any of the component reels in the feeder slots are exhausted, the machine stops and the exhausted reels have to be replenished manually by the operator. Replenishing the reels is time consuming and depends highly on the skill level and the availability of the operator. The time taken to replenish a reel depends upon two factors; time required by the feeder carriage to move back and forth from the setup position to the reload position and the time taken by the operator to load a new reel. In a high density and high volume production environment this results in an increased number of replenishments. This in turn forces the machine to stop often resulting in loss of valuable production time. Though in some cases the loss in production time due to feeder replenishment can be small when compared to the assembly time, the need for an available operator to replenish the reel may further increase the machine idle time.

Moreover, the feeder tray in the turret head type is constantly moving horizontally for component positioning and hence rules out the use of spliceable feeders and other mechanisms designed to keep the machine operating even while the reels are replenished.

Usually the number of different component types to be assigned to the feeder slots is less than the number of available slots; that is surplus or extra slots are available. The first objective of this research is to maximize the number of boards that can be made by a single allocation of components to the feeder slots and therefore reduce the idle time due to feeder replenishment. This in turn requires determining which component types are to be allocated to the surplus slots. This is called the feeder allocation problem. In this research a dynamic programming (DP) model is proposed to solve the model.

Most research conducted in this area has involved building optimization models which are usually linear programs or integer programs. These models require optimization software like Cplex, Lingo or Lindo. The biggest shortcoming of this approach is that these software products are expensive. Another disadvantage of this approach is that the managers or operators in the industry might find it difficult to operate the software and interpret the results. Therefore, the second objective of this research is to present a methodology to build user interfaces to solve such optimization problems using spreadsheets. This technique is called spreadsheet based optimization or spreadsheet modeling [2]. This is demonstrated by using the DP model.

This manuscript is organized as follows. In chapter 2, research in the area of placement machines is discussed in detail. Chapter 3 describes the problem considered for this master's thesis along with assumptions made and develops the DP formulation. In order to illustrate the working of the model an example is also presented. Chapter 4 deals

with the methodology or the steps involved in constructing a spreadsheet based interface for implementing the model. In chapter 5 the experimental results are given and discussed. Chapter 6 gives the conclusion and directions for future work.

CHAPTER 2

LITERATURE REVIEW

This chapter presents a literature review on problems in the optimization of placement machines.

Among all the steps involved in the SMT process, the placement machine is generally the bottleneck in the assembly line. Hence, any reduction in the placement times will result in considerable increase in productivity and therefore profit. According to Grunow et al [16] there are a number of reasons why placement machines do not reach their nominal placement rates. Some of these factors are the sequence of placement operations, the feeder slot assignment, the rotational speed of the turret, the varying width of the feeders, and the component retrieval. The authors also discuss the importance of sequencing of the placement operations. In a turret head placement machine the sequence and the setup are the main factors that determine the magazine movement. One way of reducing the feeder carriage movement is by multiple allocations of components to the feeder slots. In this paper, the authors have also presented a simulation based system that incorporates various types of assembly machines. According to the authors, this simulation system serves dual purposes; it can be used as a tactical production planning aid and also can be used for the kinematic analysis of the processes of the PCB assembly lines.

In the area of placement machines, several approaches have been discussed in order to improve the placement time through optimally sequencing the component placement and by proper component allocation.

In [11] Ellis et al have proposed a solution approach to determine the optimal placement sequence of components and feeder arrangement on the chip shooter machine. The authors' state that the solution approach developed can be integrated into a process planning system to reduce assembly time and improve productivity. The solution approach proposed minimizes the placement time of a turret head placement machine. The solution approach involves constructing an initial solution based on some characteristics and set of rules of the placement machine and then using a two-opt improvement algorithm to improve the initial solution. In order to evaluate the proposed solution approach, the authors have developed an estimator function for the placement time of the turret head placement machine. This estimator function has been developed using the operational characteristics of the turret head placement machine, such as the PCB table speed, turret head speed, concurrent mechanisms, and the feeder carriage speed. This solution approach resulted in placement times that are lower than those obtained from commercial software.

Kimberly et al [12] have focused on obtaining the placement time required to populate a PCB using a particular model of the chip shooter machine, namely the Fuji CP4-3. In order to do this, the authors have developed a conceptual model of the placement time estimator function for a turret head placement machine. The placement time estimator function takes into account the operational characteristics of the chip shooter machine like the PCB table speed, concurrent mechanisms, the turret head speed

and the feeder carriage speed. The estimator function also takes into account the characteristics of the components to be mounted. By means of experiments, the machine parameters that are relevant to the estimator function have also been determined. Experimental results indicate that there is no statistical difference between the estimated and the actual placement times.

Gronalt et al [15] have proposed a heuristic solution to efficiently switch components in a multiple machine and multiple board environments. The authors have decomposed the component switching problem into component set up and component assignment problems. The proposed method uses a recursive heuristic – mixed-integer linear optimization model. The heuristic consists of two stages. The first stage of the recursive heuristic is used to determine the component set up for a sequence of board types to be assembled on a single placement machine. This has been achieved by using a modification of the 'keep component needed soonest' policy. In the second stage reels of the component types are assigned to the feeder slots of the carriage. Both the stages are solved iteratively, and the solutions to the set up problem provide the basis for solving the assignment problem.

In [9] Depuy et al have developed a component allocation methodology which, for one component, allocates multiple feeder slots among multiple placement machines. The objective of their research was to minimize the work load and therefore reduce the cycle time for a printed circuit board assembly system. In order to do this, the problem has been formulated as a mixed 0-1 integer programming model that takes into account a number of operational characteristics. By this approach the authors have shown that the cycle time can be greatly improved by assigning components among multiple machines.

Crama et al [6] have focused on minimizing the make span of a PCB assembly process. The authors deal with the component retrieval problem which arises when component reels are assigned to multiple feeder slots. The component retrieval problem deals with determining the feeder slot from which components have to be retrieved for each related component placement on the board. The component retrieval problem has been formulated as a longest path problem in PERT/CPM network; that is the component retrieval problem was been formulated as a shortest path problem with side constraints. The authors have proved that this problem can be solved in polynomial time. They have also proved that a straight forward dynamic programming approach to solve the component retrieval problem with not necessarily yield an optimal solution. Therefore the authors have proposed a two phase dynamic programming approach which can be solved in polynomial time.

Crama et al [7] have focused on the planning problem of assembling different types of boards using a single line of placement machines. Three planning problems have been considered in this paper; a) finding a feeder rack assignment for each machine b) component placement sequence for each board type and each machine c) the component retrieval sequence for each machine board pair. A solution approach based on the hierarchical decomposition of the planning problem has been proposed. A heuristic has been used to solve the feeder rack assignment problem and the other sub problems have been solved using constructive heuristics and local search methods. In order to compute the feeder rack assignment without computing the placement sequence and the component retrieval plan, an estimate of the make span for each board and each machine for given feeder rack arrangement has been computed. Later these estimates have been

used as an indicator of the quality of the feeder rack assignment. The authors have also proved that, it is possible to obtain a feeder carriage assignment and a component placement sequence such that the subsequent components can be retrieved from consecutive feeder slots.

Bard et al [5] have focused on three issues a) component placement sequence b) assigning component reels to the feeder slots and c) the sequence in which components must be retrieved in case a particular component type is assigned to more that one feeder slot. All of these problems have been formulated as non linear integer programs. The authors have developed a series of algorithms to solve each of the above problems iteratively using a two step approach. The component placement sequence has been obtained using a weighted nearest neighbor traveling salesman problem heuristic. The feeder slot assignment and the component retrieval problem have been formulated as a quadratic integer program. The authors state the quadratic integer program is difficult to solve and hence have decomposed the quadratic IP into two sub problems by using Lagrangian relaxation techniques. The results of the two sub problems are then combined to obtain good solutions. Finally the current feeder carriage assignments are used to update the component placement sequence.

The other advantage of the multiple allocations of components to feeder slots is that it helps to balance the assembly line. In [3] Ammons et al have considered a component allocation problem in which there are more than one placement machines. These machines may or may not be identical. The objective of their research is to balance the assembly line. The line balancing problem has been formulated as a large scale integer programming model which takes into account multiple and non identical

machines. The model has been solved using two methods. The first method uses a heuristic based on list processing in order to solve a simplified version of the component/line balancing problem. The second method involves using a LP based branch and bound mathematical software called MINTO to solve the whole problem.

In related research Depuy et al [10] have considered the component allocation problem for coupled automatic placement machines. Their objective is to determine which machines place which component types and therefore reduce/balance the printed circuit board assembly line. In order to do this a large scale 0-1 mixed integer program has been presented. To solve the large scale integer programming model, the authors have proposed two methods. The first method deals with model simplification and data aggregation in order to reduce the size of the problem. In the second method, an integer programming heuristic has been developed by modifying a LP based branch and bound mathematical programming software.

Grunow et al [17] discuss the problem of assembly line balancing in modular placement machines in order to obtain the desired output rate. This paper focuses on three issues; a) determination of the number of feeders for each component type to be used b) assignment of component reels to modules and c) assignment of placement operations to modules. The third problem is referred to as the workload balancing problem. In order to solve these problems the authors have proposed an integer program and two heuristic solution procedures. In both the heuristics the first stage is used to generate a feasible solution satisfying the constraint of limited component magazine capacity at each module of the placement machine. The second stage deals with balancing the workload of each module in order to minimize the cycle time. In order to balance the work load the authors

have presented two alternatives. The first alternative uses priority rules in order to remove the bottlenecks in the workload. This is achieved by reassigning some of the assembly operations to other modules of the machine. The second alternative determines the optimal solution by solving an IP.

Zijm and Harten [23] discuss the design of a process planning system of a printed circuit assembly line, which is hierarchically structured. The authors have considered an assembly line that has a pipette head as a placement machine. The objective of the authors is to balance the workload at different machines and thereby minimize cycle time. This is achieved by determining the number of reels of each component type to be used. The authors have also shown that the process planning problem can be decomposed into a series of hierarchically coupled sub problems and each of these problems are of combinatorial structure. Models and solution techniques for solving the sub problems are also discussed.

Neammanee and Randhawa [20] have focused on the problem of assigning boards to production lines with the objective of minimizing the assembly time. The authors state that the board assignment and component allocation have to be performed concurrently as the set up times are sequence dependent. In order to solve this problem an integrated methodology having seven phases has been proposed. The seven phases are

a) Grouping the printed circuit boards.

b) Family decomposition.

c) Sequencing of sub families.

d) Keep tool needed soonest (KTNS) procedure.

e) Determination of component set up.

f) Component allocation.

g) Board assignment.

The effect of two parameters, capacity of the feeders and threshold value (indicates the effectiveness of joining a component type to a component group), on the proposed solution procedure has also been studied. The outcome of the study indicates that the capacity of the feeders influences the total workload imbalance but does not have any effect on the total assembly time. The threshold value was found to have a significant effect on the total make span. The interactions of the threshold value, variations in requirements of PCB's and component usage, also have a significant effect on the global make span. Moreover the authors have also shown that their proposed methodology can solve large scale problems quickly and efficiently.

Francis and Horak [13] have proposed an integer program and a bisection algorithm for determining number of reels of each component type to be used. This model aims to maximize the number of boards that can be assembled before the first reel runs out.

Ahmadhi et al [1] have considered a pick and place machine with two feeder carriers for the delivery of components to the placement head. The issues that arise in the optimization of pick and place machines with dual carriers are a) the number of reels of each component type to be allocated b) the assignment of each reel to the carrier c) the alternating pick and place sequence between each carrier and the PCB d) the position of each reel on the assigned carrier. In this paper the authors have focused on two issues; component allocation and partitioning. In component allocation, the objective is to determine the number of reels of each component type to be used. In order to do this the

authors have developed an integer program model that maximizes the number of boards that can be made by a single allocation of components to the carriers, without any replenishment. Once the number of reels of each component type to be used is determined, the objective of the partitioning problem is to determine on which two feeder carriers each component reel has to be assigned. The author states that the partitioning problem is very important as it greatly affects the throughput time. The partitioning problem has been formulated as an integer program that uses the output of the component allocation problem to minimize the dead time. Dead time is the down time incurred due to operation imbalance, excess rotation of each head and nozzle changes. Moreover for each problem three different scenarios have been considered which vary depending on the fixtures used for the delivery of components and the pick sequence.

Ho and Ji [18] have focused on two placement problems of a turret head placement machine; a) arrangement of component types to the slots in the feeder carriage b) sequence of the pick and place operations. The authors have proposed a hybrid genetic algorithm to determine the optimal placement sequence and the feeder arrangement simultaneously. The hybrid genetic algorithm uses different search heuristics such as, the nearest neighbor heuristic, the 2 opt heuristic and a new heuristic called iterated swap procedure. In the algorithm, the chromosome is represented by means of a two link structure; the first link represents the component placement sequence and the second link denotes the feeder arrangement. The initial population for the first link is generated by means of the nearest neighbor heuristic and randomly for the second link. Then an iterative swap procedure is applied to the first link and a 2 opt heuristic is applied to the second heuristic. This is then followed by selection, modified crossover and heuristic

mutation operations. The authors have also shown that the proposed hybrid genetic algorithm gives better results than a simple genetic algorithm.

In [21] Ramasamy et al have developed a feeder carriage replenishment policy for a turret head placement machine. The objective of the authors is to reduce the idle time of the turret head placement machine due to frequent replenishments by using multiple allocations of components to the feeder carriage slots and effectively synchronizing the feeder exhausts. A mixed 0-1 integer program has been presented to do the same. The proposed model differs from other models in the sense, if a particular component is assigned to more than one slot, than the demand for that component type is split among all the slots. That is an additional problem of determining how many components of each component type has to be picked from each slot arises. Since the model is based on the fragile synchronization of the feeder exhaust any variation in reel sizes can disrupt the synchronization therefore increasing the number of replenishments. In order to counter the variation in reel sizes due to supplier allowances, a policy called the Fixed Replenishment (FR) policy has been proposed.

CHAPTER 3

PROBLEM FORMULATION

As described in chapter 1, each time the feeder slots of the turret head placement machine runs out of components, the machine stops and waits to be replenished. The downtime incurred due to frequent replenishments can be considerably high. This chapter deals with the first objective described in chapter 1; maximize the number of boards that can be made by a single allocation of components to the feeder slots and therefore reduce the idle time due to feeder replenishment. This in turn requires determining which component types are to be allocated to the surplus slots. In this chapter a dynamic programming (DP) model is presented along with the assumptions and notations. The model is also illustrated with a small example.

3.1 Assumptions

Some of the assumptions that have been made while formulating the DP model are

1. Each component type reel can be accommodated in a single slot; that is the width of the reels are lesser than the width of the feeder slots
2. All the surplus slots have to be used
3. The reel sizes and demand are already known; they are deterministic

4. There are no variations in reel sizes

5. Each slot has one mandatory reel

Notice that assumption 2 is made to reduce the number of computations since it is always better to use all the surplus slots.

3.2 Notations

T Number of different component types = Number of stages

t Index for the component types ; t = 1 to T

S Total number of surplus slots

r_t Reel size of component type t

d_t Demand of component type t in each board

X_t Number of surplus slots available to assign to component type t, t+1, t+2, …T; $X_t = 0$ to S

Y_t Number of extra reels of component type t to assign to the surplus slots; decision variable

$R_t(Y_t)$ Number of boards that can be made using Y_t reels of component type t; Y_t = 0 to X_t

$F_t(X_t)$ Maximum number of boards that can be made stages t through T given that current state is X_t

3.3 Model formulation

Dynamic programming (DP) is a method of decomposing problems that are hard to solve, into smaller problems that can be solved easily. In order for a problem to be

formulated as a DP, it has to be a Markovian process, that is, the future must depend only on the present.

In order to formulate a problem as a DP, the stages of the system have to be identified. In our problem, the different component types, t = 1 to T is considered to be the different stages of the system. By this manner the sequential property is induced into the system. Next, the state of the system has to be defined in such a way that it completely describes the system. In our problem, the decision to be made is determining which components to assign to the surplus slots. The state of the system at each stage is defined as the number of surplus slots at that stage. The decision to be made in each stage t when the system is in the state X_t, is how many reels, Y_t, of the component type t have to be assigned to the surplus slots. Once a decision (Y_t) is made, a return $R_t(Y_t)$ is generated. The return is the number of boards that can be made by using Y_t+1 reels of component type t. Based on the decision made, the system is transformed into a new state at the next stage. Because of the way the problem is set up, it can be clearly noted that the problem is Markovian.

The objective is to maximize the total return (number of boards) over all the stages. Hence the problem can be stated as

$$F_t(X_t) = \max_{0 \le Y_t \le X_t} \{Min[R_t(Y_t), F_{t+1}(X_t - Y_t)]\} \qquad \forall t \neq T \qquad (1)$$

$$\text{and } X_t = 0, 1 \ldots S$$

$$F_T(X_T) = R_T(X_T) \qquad X_T = 0, 1 \ldots S \qquad (2)$$

Where

$$R_t(Y_t) = \frac{r_t * (Y_t + 1)}{d_t} \qquad \forall t = 1, 2 \ldots T \quad \forall Y_t = 0, 1 \ldots X_t \qquad (3)$$

The above formulation represents a backward recursive dynamic programming approach. Equation (1) determines the maximum number of boards that can be made over all the stages. Equation (2) determines the return for the stage $t = T$. We have $Y_t = X_t$ because of the assumption that all surplus slots available will have to be used. Equation (3) determines the number of boards that can be made using Y_t+1 reels of component type t. The number of boards that can me made can be determined by multiplying the reel size of a component and the number of reels used and then dividing this amount by the demand for that component type. In equation (3) $Y_t + 1$ is used instead of Y_t because Y_t represents the number of extra reels to be used and due to the fact that one reel of each component type has already been assigned to the mandatory slots.

To begin with, the number of boards that can be made for the states in the last stage, $t = T$, is determined using equations (2) and (3). Then for the states in the stage T-1, the number of boards that can be made is determined using equation (1). Equation (1) uses the information obtained from the previous stage and the return of the current stage to determine the maximum number of boards that can be made for all the states in that stage. That is the return for a particular state and a stage t represents the best result for that stage t and the previous stage t+1. This kind of backward recursion is continued until stage 1 and the maximum number of boards obtained in stage $t = 1$ represents the optimal result over all the stages. In order to get the optimal allocation plan the results obtained in stage $t = 1$ would have to be back tracked.

A diagrammatic representation of the DP model with four different component types and three surplus slots is shown in Figure 3.1.

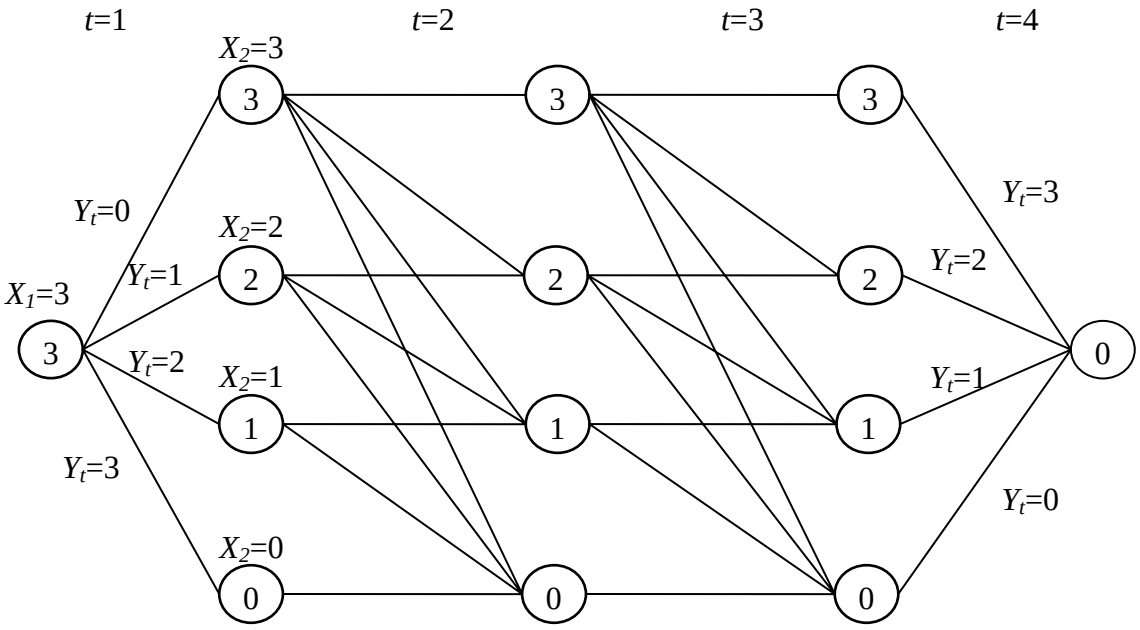

Figure 3.1: DP representation

In Figure 3.1, the numbers inside the circles represents the number of surplus slots (states) available. At the end of the stage t = 4 the number of surplus slots available is zero because of the assumption that in the last stage all the available slots have to be used. X_t represents the states and Y_t represents the decisions taken in each state.

3.4 Example

In order to further illustrate the working of the dynamic programming model an example is presented below. The example consists of four different component types and three surplus slots. The data for the example problem are given in Table 3.1.

Table 3.1: Example data

Component type (t)	Reel size (r_t)	Demand (d_t)
1	3000	20
2	2000	10
3	1000	30
4	2000	10

Since this is a backward recursion, the DP starts from stage t = 4. This information is given in Tables 3.2, 3.3, 3.4, and 3.5.

Table 3.2: Stage t = 4

X_t	Y_t	$R_t(Y_t)$	$F_t(Y_t)$
3	3	800	800
2	2	600	600
1	1	400	400
0	0	200	200

Table 3.3: Stage t = 3

X_t	Y_t	$R_t(Y_t)$ (3)	$F_{t+1}(X_t-Y_t)$ (4)	Min (3),(4)	$F_t(X_t)$
3	0	33	800	33	133
	1	66	600	66	
	2	100	400	100	
	3	133	200	133	
2	0	33	600	33	100
	1	66	400	66	
	2	100	200	100	
1	0	33	400	33	66
	1	66	200	66	
0	0	33	200	33	33

Table 3.4: Stage $t = 2$

X_t	Y_t	$R_t(Y_t)$ (3)	$F_{t+1}(X_t-Y_t)$ (4)	Min (3),(4)	$F_t(X_t)$
3	0	200	133	133	133
	1	400	100	100	
	2	600	66	66	
	3	800	33	33	
2	0	200	100	100	100
	1	400	66	66	
	2	600	33	33	
1	0	200	66	66	66
	1	400	33	33	
0	0	200	33	33	33

Table 3.5: Stage $t = 1$

X_t	Y_t	$R_t(Y_t)$ (3)	$F_{t+1}(X_t-Y_t)$ (4)	Min (3),(4)	$F_t(X_t)$
3	0	150	133	133	133
	1	300	100	100	100
	2	450	66	66	66
	3	600	33	33	33

Now in order to get the optimal allocation plan the results from Table 3.5 have to be back tracked. From the Table 3.5 it can be seen that the best result (maximum number of boards that can be manufactured by a single allocation of component type reels) is 133, which corresponds to $X_1 = 3$ and $Y_1 = 0$. This results in the system transforming to state 3 in stage 2 for which best result is 133, for $Y_2 = 0$. The system now transforms to state 3 in stage 3 for which the best result is 133 for $Y_3 = 3$. Finally the system transforms to state 0 in stage 4, corresponding to $Y_4 = 0$. In order to further illustrate the above sequence of events Figure 3.2 is presented.

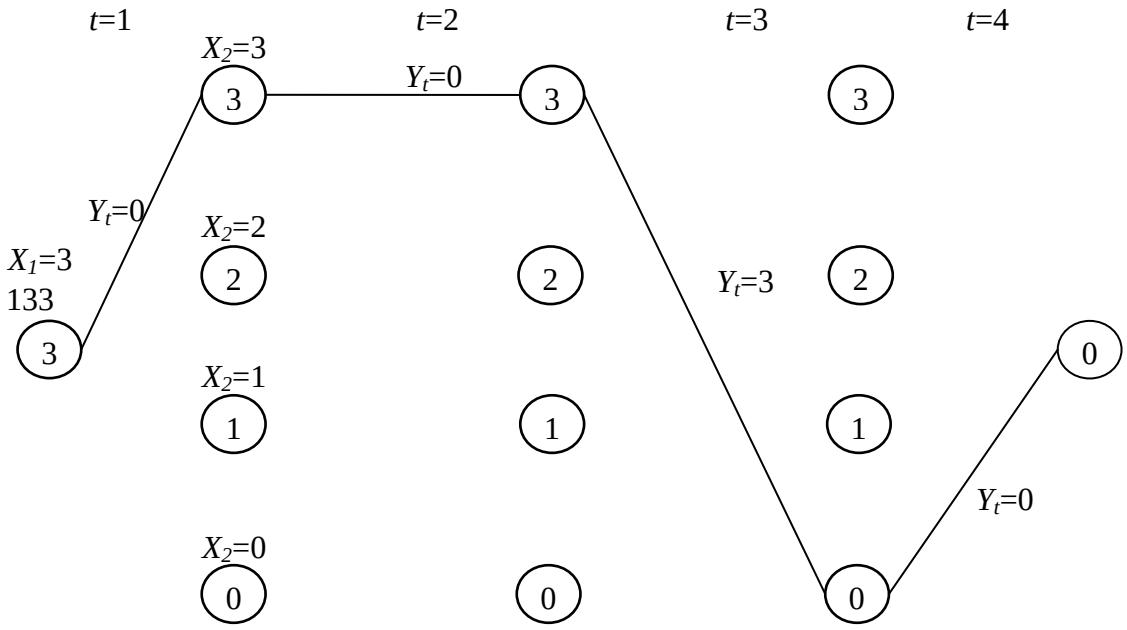

Figure 3.2: Optimal solution

The above diagram can be interpreted as follows. The maximum number of boards that can me made by a single allocation of component reels is 133. In order to manufacture 133 boards, 0 extra reels (Y_1=0) of component type 1(t = 1) are used. This results in three surplus slots available for component type 2 (t =2). Now zero extra reels (Y_2 = 0) of component type 2 (t = 2) are used. This results in three surplus slots available for component type 3 (t = 3). Now all the 3 extra reels (Y_3 = 3) of component type 3 are assigned to all the three surplus slots resulting in no surplus slot available for component type 4 (t = 4). (That is 0 extra reels of component type 4 are used).

The example discussed above has 4 stages and 13 states. The number of stages for any problem is equal to the number of different component types. The total number of states for any problem can be obtained by using equation (4).

$$\text{Total number of states} = S \times (T - 1) + T \qquad (4)$$

For example, by using equation (4), the number of states for a problem with 10 different component types (T = 10) and 5 surplus slots (S = 5), is 55. The complexity of the dynamic programming model proposed in this thesis can be evaluated by multiplying the number of cost calculations and the time for one cost calculation. Let n represent the number of cost calculations and z the time for one cost calculation. Then the number of cost calculations n can be obtained from equation (5).

$$n = 2 \times (S + 1) + \frac{(S + 1) \times (S + 2) \times (T - 2)}{2} \tag{5}$$

For example, by using equation (5), the number of cost calculations for the example discussed is section 3.4 is 28. The time for one cost calculation, z, for the DP model proposed in this thesis can be obtained from equation (6).

$$z = e \times l \tag{6}$$

where e is the number of elementary operations (addition, subtraction, multiplication, division and comparison) performed for one cost calculation and l is the time taken by the computer for performing one elementary calculation. For the DP model proposed in this thesis, 3 elementary operations are performed to obtain one cost calculation and hence e is equal to 3. l is a constant and depends upon the computer hardware.

The complexity of the proposed DP model can be evaluated by multiplying n and z. If r represents the maximum of S and T, it can be observed that since z is a constant, the complexity (worst case) of the model is $O(r^3)$.

3.5 Alternative solution approaches

In the following section two alternative solutions to solve the feeder allocation problem is presented. The first solution approach is the IP formulation proposed by Ahmadhi et al [1]. The second method is a sequential search method.

3.5.1 IP formulation

Let b represent the number of boards that can be made before the first reel runs out. Then the IP formulation for the feeder allocation problem is given as

$$Max\, b \tag{7}$$

S.T

$$b \le \frac{r_t \times (Y_t + 1)}{d_t} \qquad \forall\, t = 1, 2..T \tag{8}$$

$$0 \le Y_t \le S \qquad \forall\, t = 1, 2..T \tag{9}$$

$$\sum_{t=1}^{T} Y_t \le S \tag{10}$$

Y_t and b are integer

3.5.2 Sequential search approach

In this approach the number of boards that can be made without using any extra slot is computed for each of the component types. The minimum value of the number of boards indicates which component type would run out first. That particular component type is assigned to the first surplus slot. Next the number of boards that can be made using each of component types including the component type assigned to the surplus slot

is again computed. The minimum value of the boards would indicate which component type would run out next. That component type is assigned to the second surplus slot. This process is continued until all surplus slots are assigned. This procedure also gives the optimal solutions for the feeder allocation problem. The pseudo-code of this procedure is given below.

For each surplus slot (S) Do

$$q = \underset{t \in T}{\text{Argmin}}\left(\frac{r_t \times (Y_t + 1)}{d_t}\right)$$

$$Y_q = Y_q + 1$$

Endfor

CHAPTER 4

IMPLEMENTATION METHODOLOGY

This chapter deals with the second objective of this thesis. In the following sections a methodology of solving optimization problems using spreadsheets (spreadsheet based optimization) is presented. This is illustrated by building a user interface for the feeder allocation problem described in the previous section. To illustrate the methodology of spreadsheet based optimization using VBA, the DP model and the sequential search approach have been coded. The code of the DP model is presented in Appendix A and the code of the sequential search solution method in Appendix B. Furthermore to explain the use of the Excel solver, the IP model is implemented.

4.1 Need for spreadsheet based optimization

Optimization problems are generally formulated as linear programming models or integer programming models. These models are commonly implemented using optimization packages such as Cplex, Lingo, and Lindo. The biggest disadvantage of this approach is that these software packages are expensive. Hence it is imperative that optimization problems be implemented keeping the cost involved in mind. This has been the main motivation for using spreadsheets in this research. Spreadsheets like Excel are

comparatively cheaper than mathematical optimization packages. Though an IP formulation for the reel allocation problem exists, a dynamic programming model for the same has been formulated. The advantage of using the DP model is that it can be easily implemented using any of the available programming languages.

In this research a user interface for the reel allocation problem has been built using Microsoft Excel. In today's world almost every organization uses spreadsheets. The main reason for using Excel as an interface is that it is very versatile, simple to use, and requires little or no training. Apart from the feeder allocation problem, Excel can be used to solve a number of decision making problems [2]. It has many integrated functions like the Solver add-in and Visual Basic for application (VBA). For example, an IP formulation for the reel allocation problem can be solved using the Solver add- in or the DP model can be coded using VBA. The spreadsheets can be used as user interfaces. Moreover, these spreadsheet based interfaces, make managing data and interpreting results easy for the operators and the managers. This versatility and simplicity of Excel that have been the main reasons for spreadsheet based optimization gaining a lot of popularity of late. The difference between spreadsheet based optimization and non spreadsheet based optimization is illustrated by the Figures 4.1, 4.2 and 4.3.

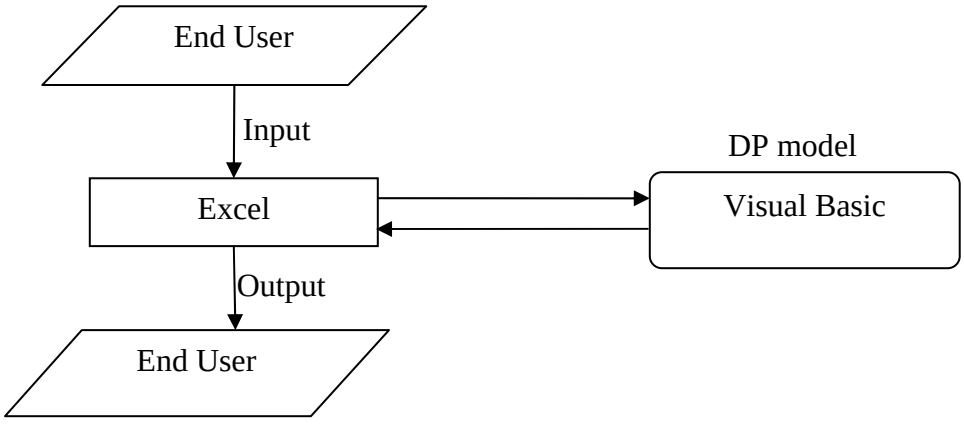

Figure 4.1: Spread sheet based optimization 1

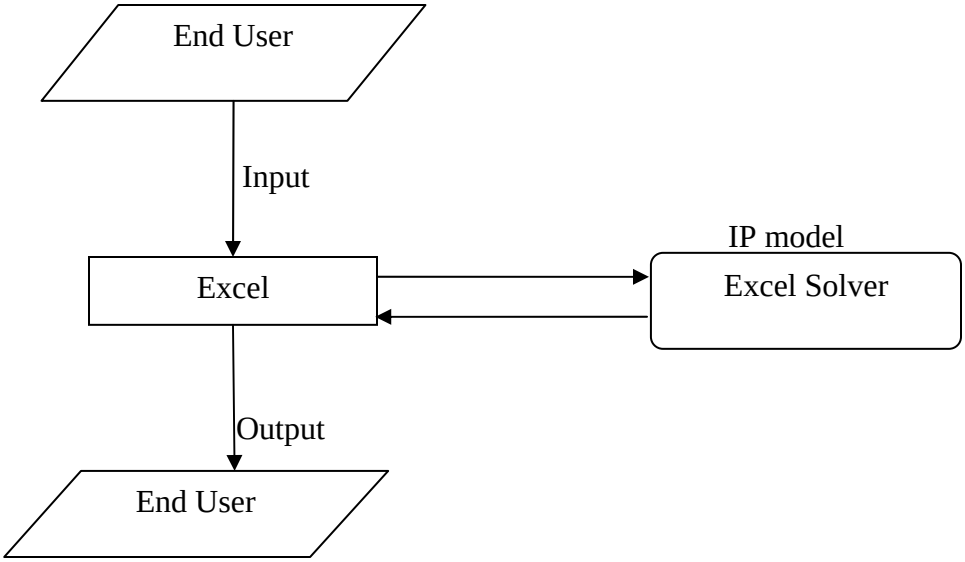

Figure 4.2: Spread sheet based optimization 2

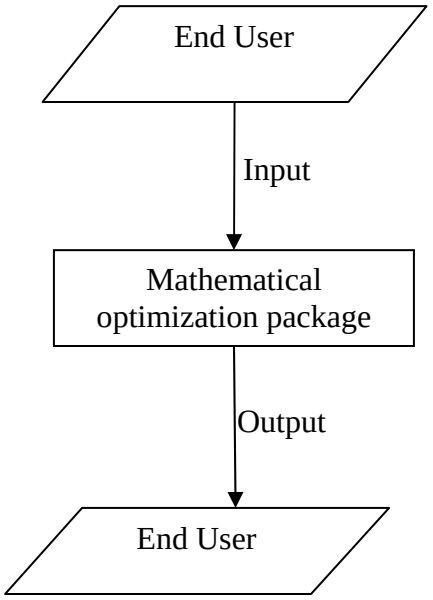

Figure 4.3: Non spread sheet based optimization

4.2 Steps involved in spreadsheet based optimization

In the following section, potential steps involved in spreadsheet based optimization are discussed. In the methodology proposed in this thesis, there are five basic steps in spreadsheet based optimization ; identifying end user requirements/defining the objectives, designing the worksheets, building the model, integrating the worksheets, and testing the model. A diagrammatic representation of the steps involved in Excel based optimization is shown in Figure 4.4.

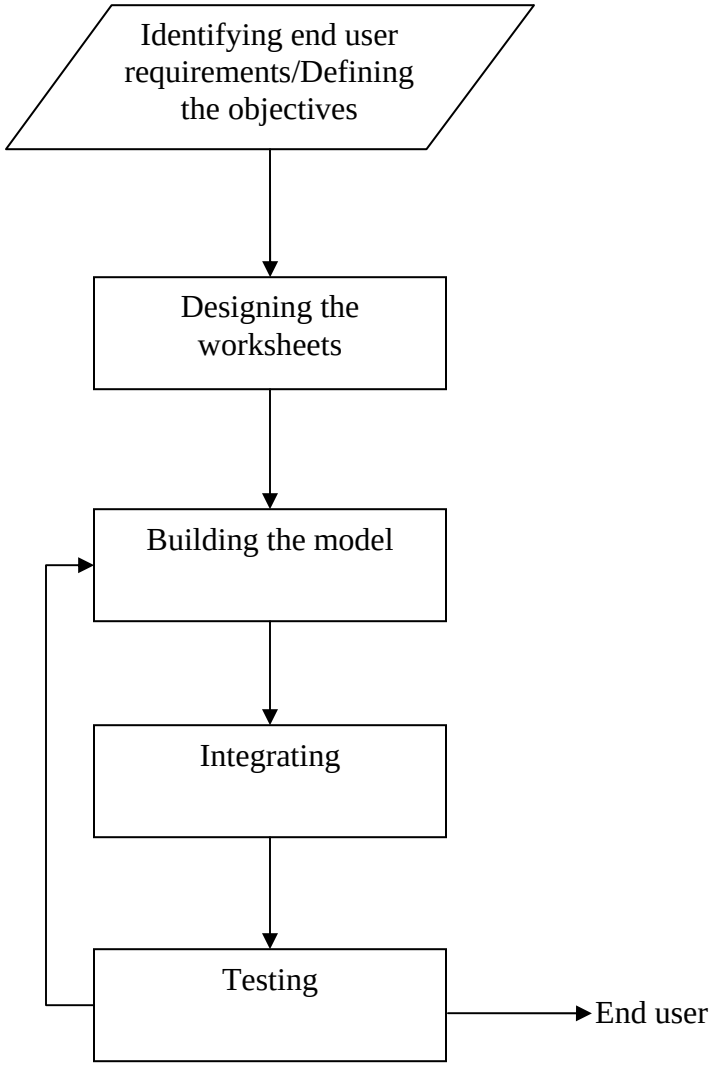

Figure 4.4: Steps involved in spreadsheet based optimization

4.2.1 Identifying end user requirements/Defining the objectives

The first step is identifying the requirements of the end user. This is a very crucial step owing to the fact that any errors committed in this stage would reflect in one of the latter stages and the entire process would have to be started from the beginning. This step involves meeting with the end users and finding out what exactly they want. Once the user requirements are identified, the problem has to be clearly understood and defined. In the reel allocation problem considered in this thesis, the user would want to determine how many reels of each component type should be used. This would be the problem statement or definition. After defining the problem, the method by which the problem would have to be solved should be determined. Since the optimization problem would have to be solved using Excel there are two options of doing this as described by Figure 4.1 and Figure 4.2. In this thesis, the decision was made to use DP to solve the model. Once the type of model to be used is decided upon, input data and output required for the model will have to be identified. While doing this it is imperative to keep the end user in mind and keep things simple. It is very important to identify only the relevant data. This is one of primary principles of spreadsheet based modeling [8]. Unnecessary information or data can be avoided. For example, the input data required for the reel allocation problem are the number of different component types, the number of surplus slots available, the demand per board of each component type and the number of components in each reel (reel size). The output generated by the model is the number of reels of each component type to be used. Finally the assumptions made will have to be specified. The steps described above are represented by means of Figure 4.5.

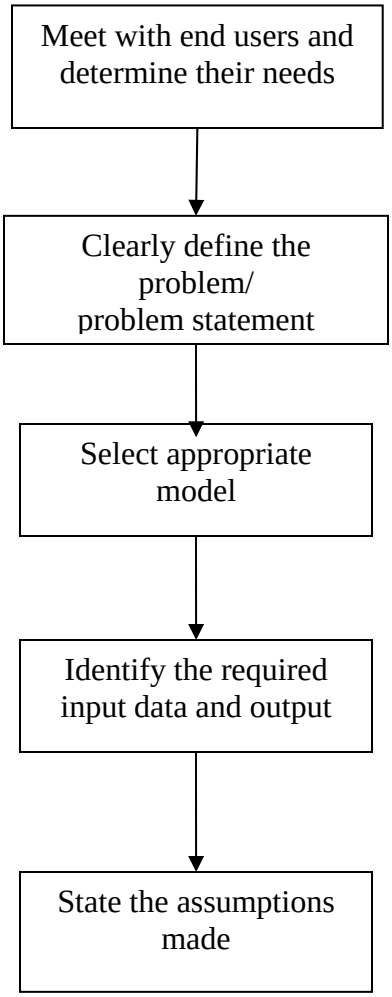

Figure 4.5: Identifying end user requirements

4.2.2 Designing the worksheets

The second step in spreadsheet based optimization is the design of worksheets. This step requires a lot of planning and foresight as it determines the entire structure of the user interface. The importance of a good design cannot be overstated as it determines the ease with which end users can work with the interface.

The first step in designing the worksheets is grouping like data together. This will usually result in three categories, input, output and calculation worksheets. The idea

behind doing this is to design separate worksheets for input, calculation and output so as to not to mix data and eliminate the possibility of any errors or confusion.

After this is done, in the second step the introductory sheet can be designed, which gives the name of the model, briefly describes the purpose of the model, and gives the name(s) of the creators of the model. An introductory sheet designed for the reel allocation problem is shown below in Figure 4.6.

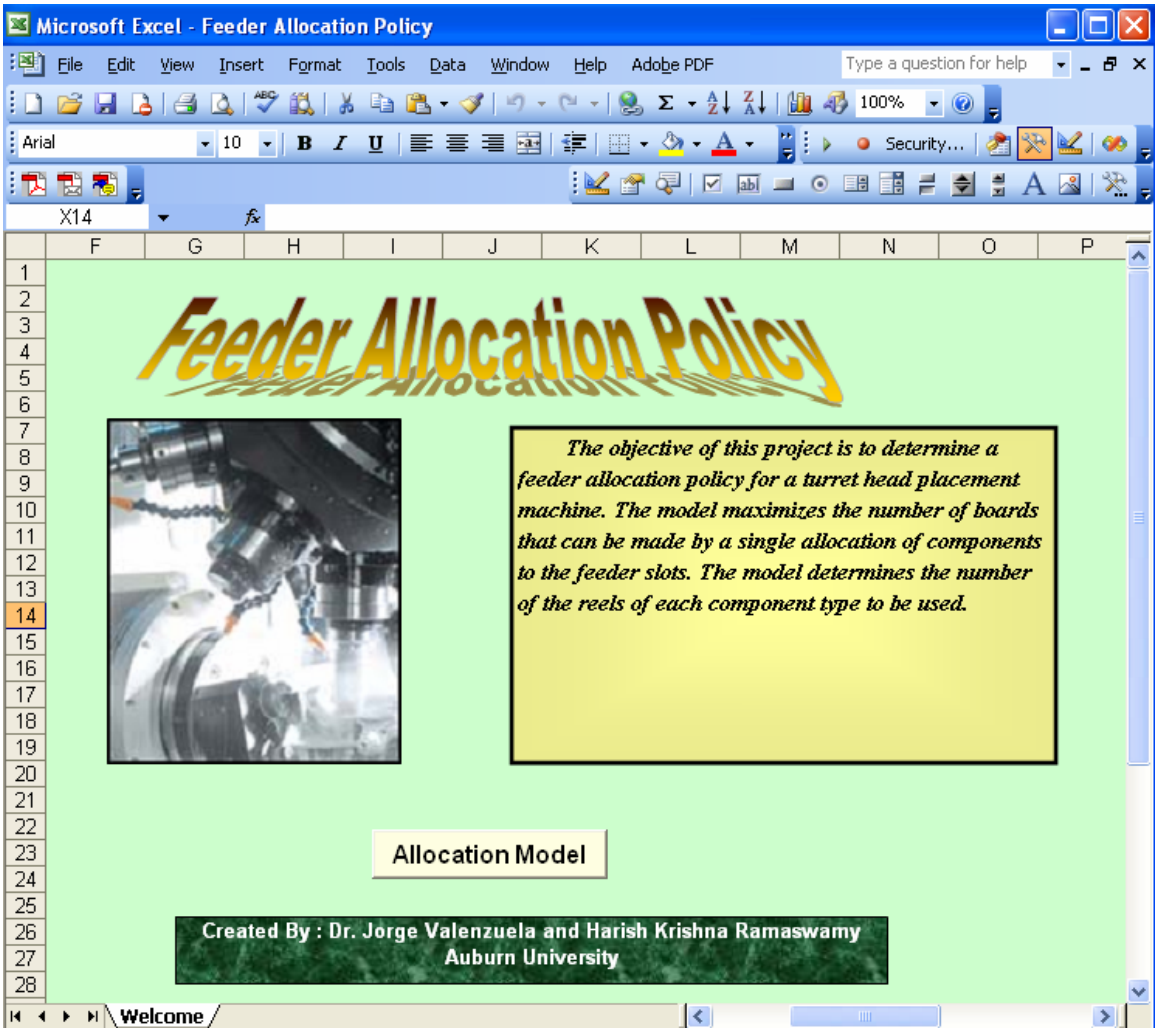

Figure 4.6: Example of an introductory sheet

The third step is designing the input sheet, calculation sheet and result sheet. It is imperative to note that a calculation sheet might not be needed if all the calculations are

performed using VBA. But if Excel solver is being used, a calculation sheet will be needed.

In Excel when new sheets are created, default names for the sheets are automatically assigned. These sheets can be given a name, such that it conveys to the user what the sheets is being used for. For example input sheet, results sheet etc. The methodology followed for designing the input sheet is described below. If possible the input page should be used to take input from the end user and not for performing calculations. The beginning of the input page should list out the steps the end user would have to follow. This is done in order to help the user navigate through the input sheet with ease. An example of this is shown in Figure 4.7.

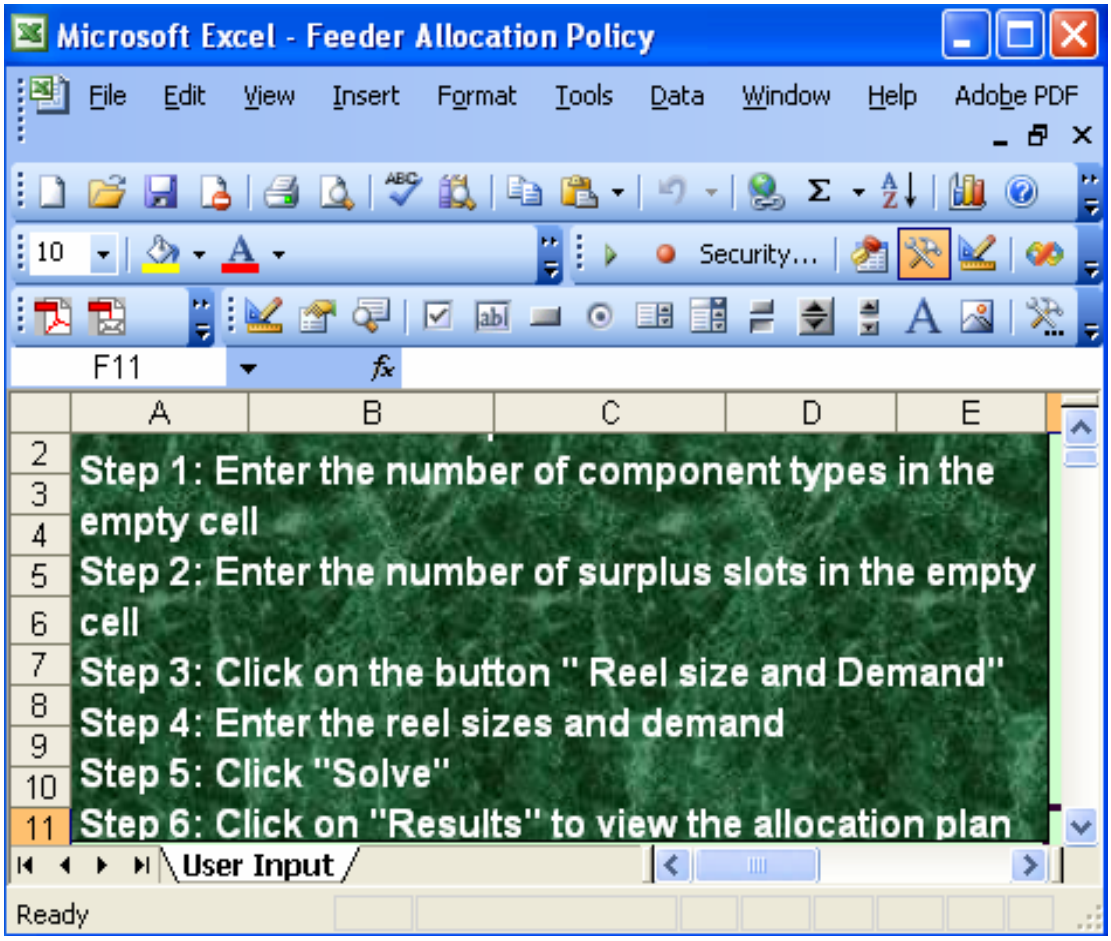

Figure 4.7: Example (1) of input sheet

Once the procedure is specified, the steps outlined in the procedure can be laid out one below the other in a logical manner. An example of this is shown in Figure 4.8.

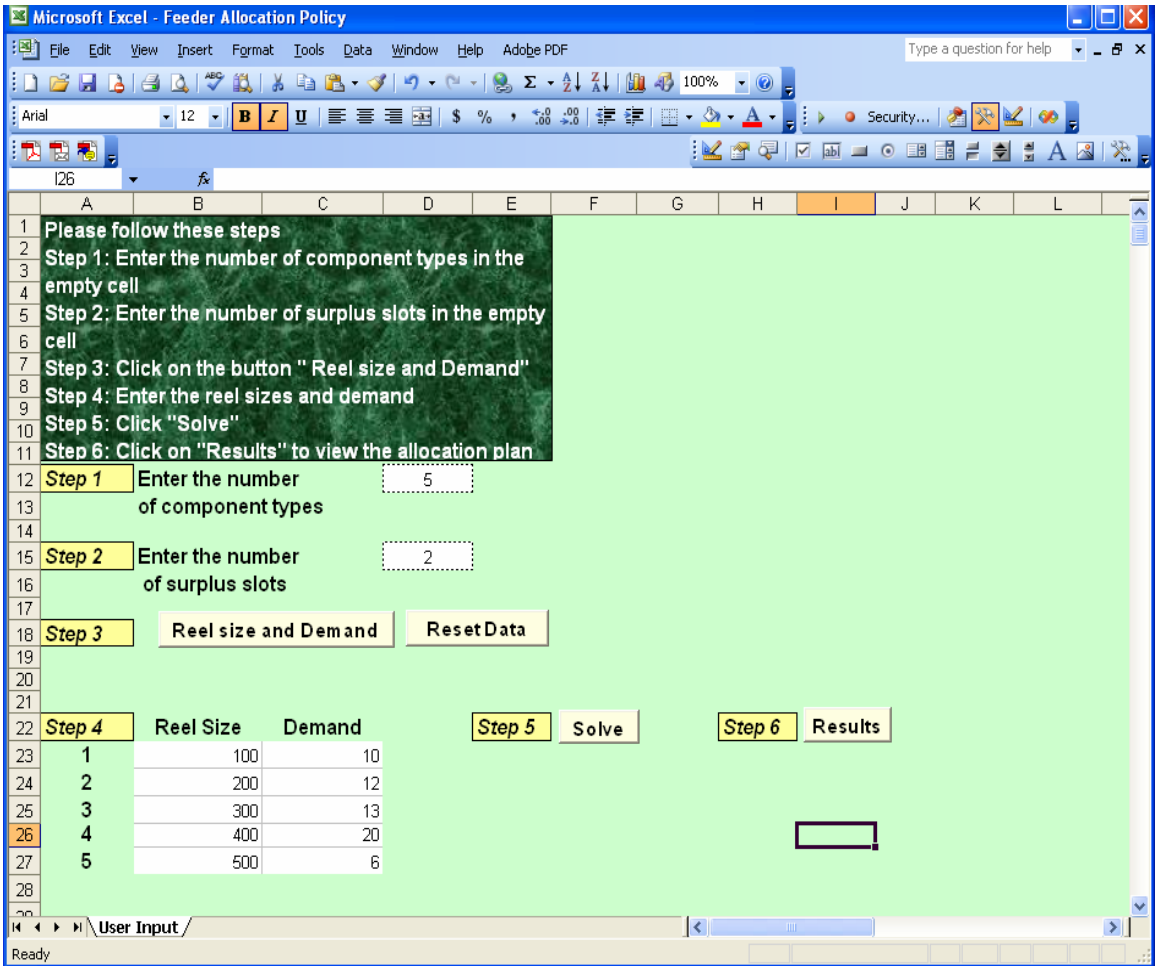

Figure 4.8: Example (2) of input sheet

It can be seen from Figure 4.8 that, the number of component types and surplus slots is asked from the user before asking for the reel sizes and demand values.

Sometimes the data required from the user may be dynamic, that is it changes depending upon some parameter. For example, in the reel allocation problem, number of reel size values and demand values that has to obtained from the user depends upon the number of different component types. Hence the number of reel sizes and demand values are dynamic. Obtaining this kind of dynamic data can be made possible by use of “buttons”. Buttons are objects that are provided by Visual Basic. These buttons can be used on any Excel sheet. Each button can be programmed (that is a Visual Basic code can

be written) to execute some action. That is why they are called event triggered procedures. This is illustrated by Figures 4.9 and 4.10.

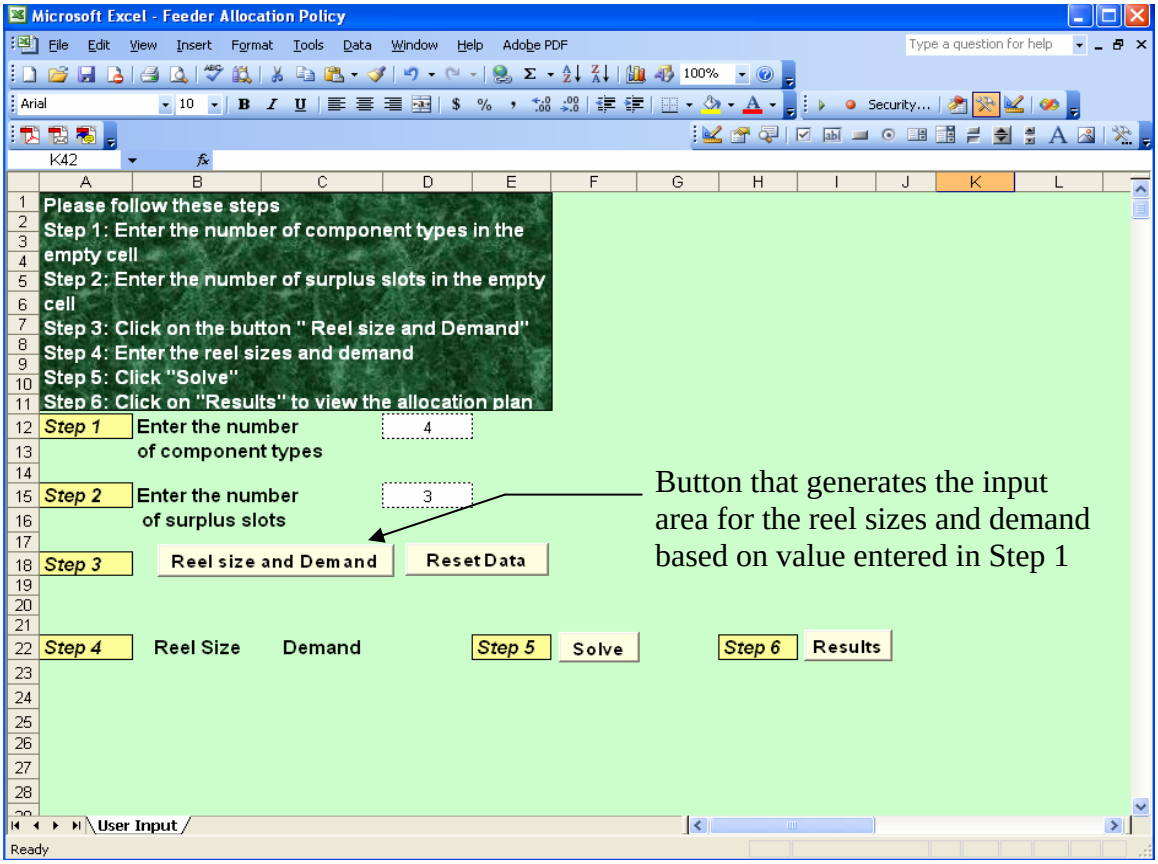

Figure 4.9: Example (1) of use of buttons

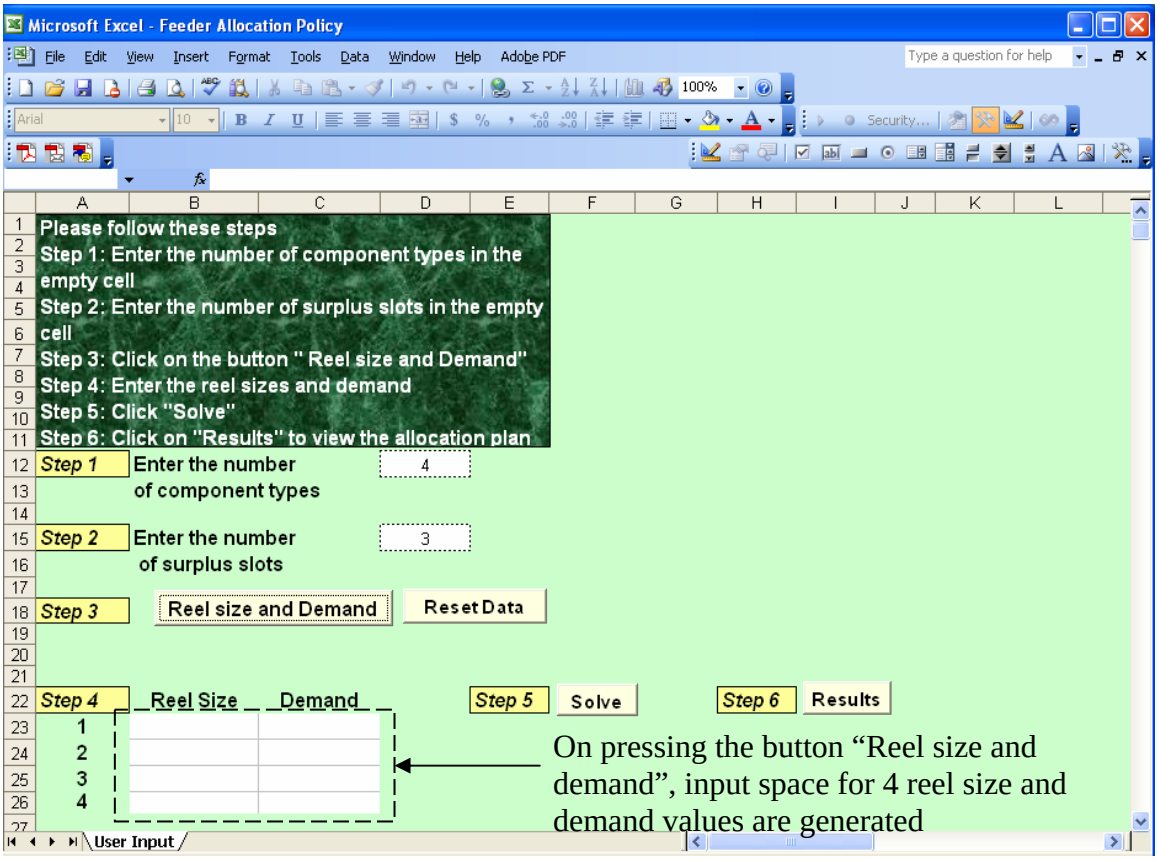

Figure 4.10: Example (2) of use of buttons

It can be seen from Figures 4.9 and 4.10 that once, the number of component types is entered in Step 1 and the button “Reel size and demand” is pressed, the required input area for entering the reel sizes and demand values is generated.

The methodology described in the above sections can be used to design the results sheet and calculation sheet. Figure 4.11 shows the design of the results sheet.

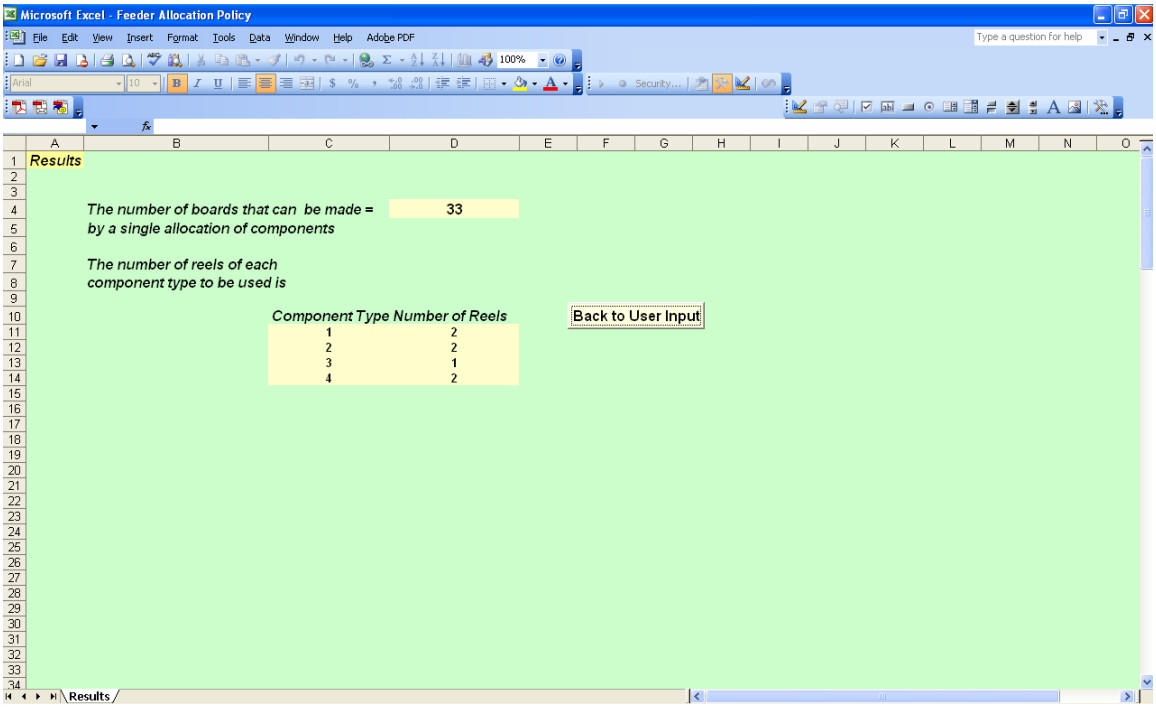

Figure 4.11: Example of results sheet

It is imperative to note that, this research emphasizes on solving the reel allocation problem using DP. Hence a calculation sheet is not used. If the problem is solved using the solver, then a calculation sheet will have to be designed. When designing the calculation sheet, it would be better if separate modules are created for the decision variables, objective function and constraints [4] .An example of a calculation sheet is shown in Figure 4.12.

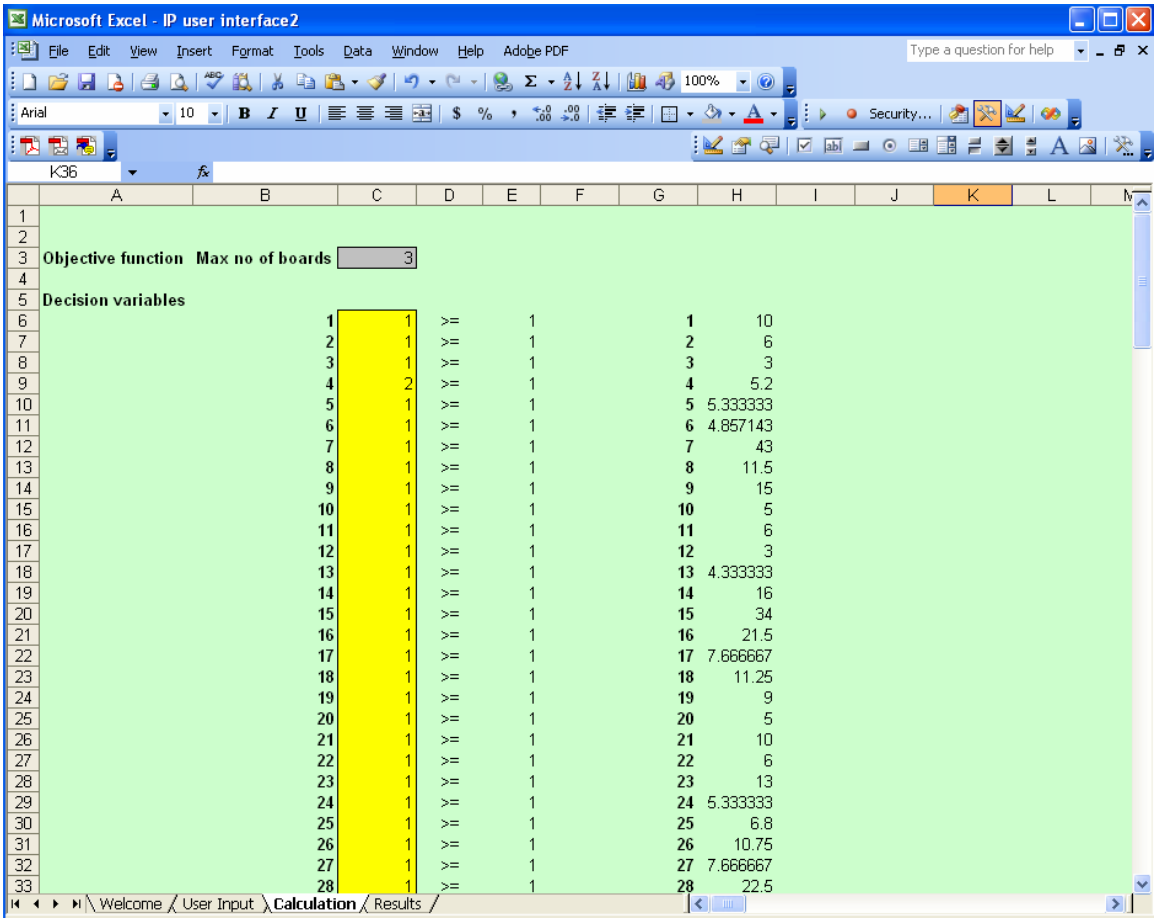

Figure 4.12: Example of calculation sheet

There are few other design aspects that relate to design of any sheet. In order to be systematic the sheets can be designed in a sequential manner; that is the user input sheet can be designed first, then the calculation sheet, and finally the result sheet. The reasoning behind this is that, while solving any problem the input is collected first, then calculations are performed and finally results are presented. Colors can be used to help the users navigate through the worksheet. For example all the input cells can be in one color and all cells in which output is displayed in another color. All the cells except those which take input from the user or displays results can be hidden. This again can be done by using colors. If the user commits any mistake while entering data or wants to resolve the problem with different data, an option of automatically resetting the data or clearing

the data can be provided. This can be done by the use of buttons. An example of this is shown in Figures 4.13 and 4.14.

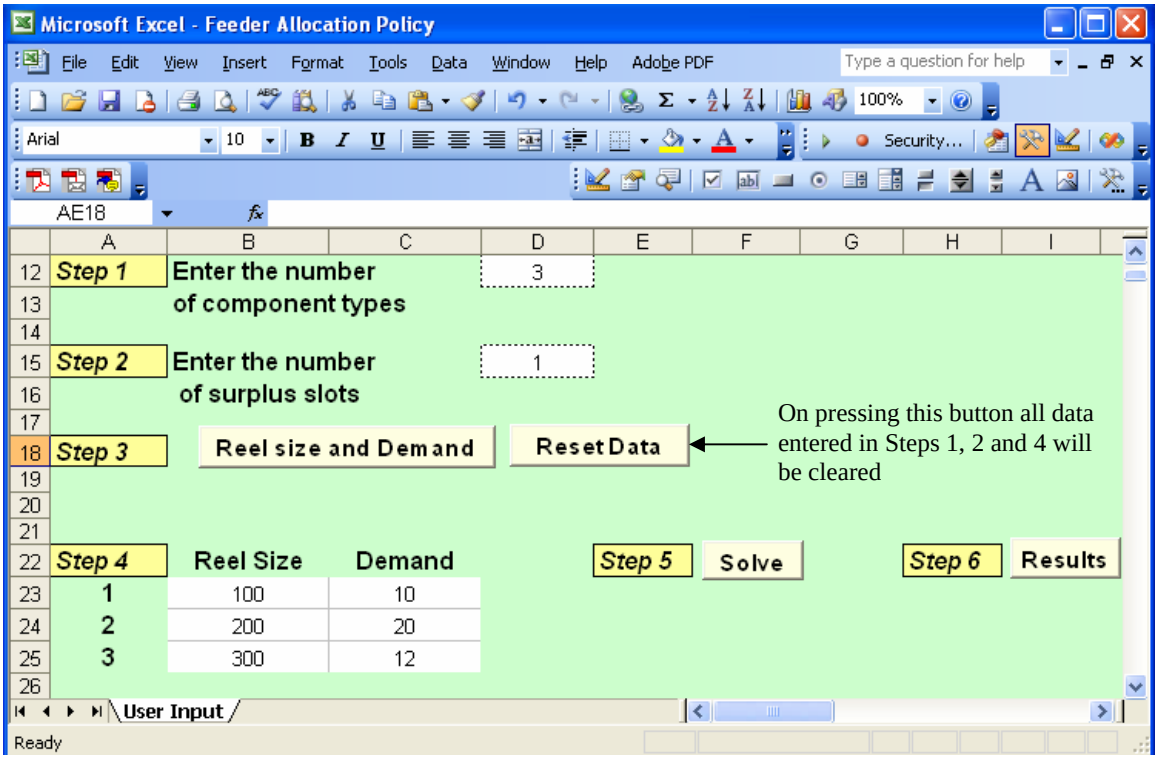

Figure 4.13: Example (1) of resetting data

As shown in Figure 4.13 let us assume the user initially wants to solve the problem for 3 component types, but then decides there are 4 different component types. On pressing the “Reset Data” button all the previously entered data will be lost as shown in Figure 4.14.

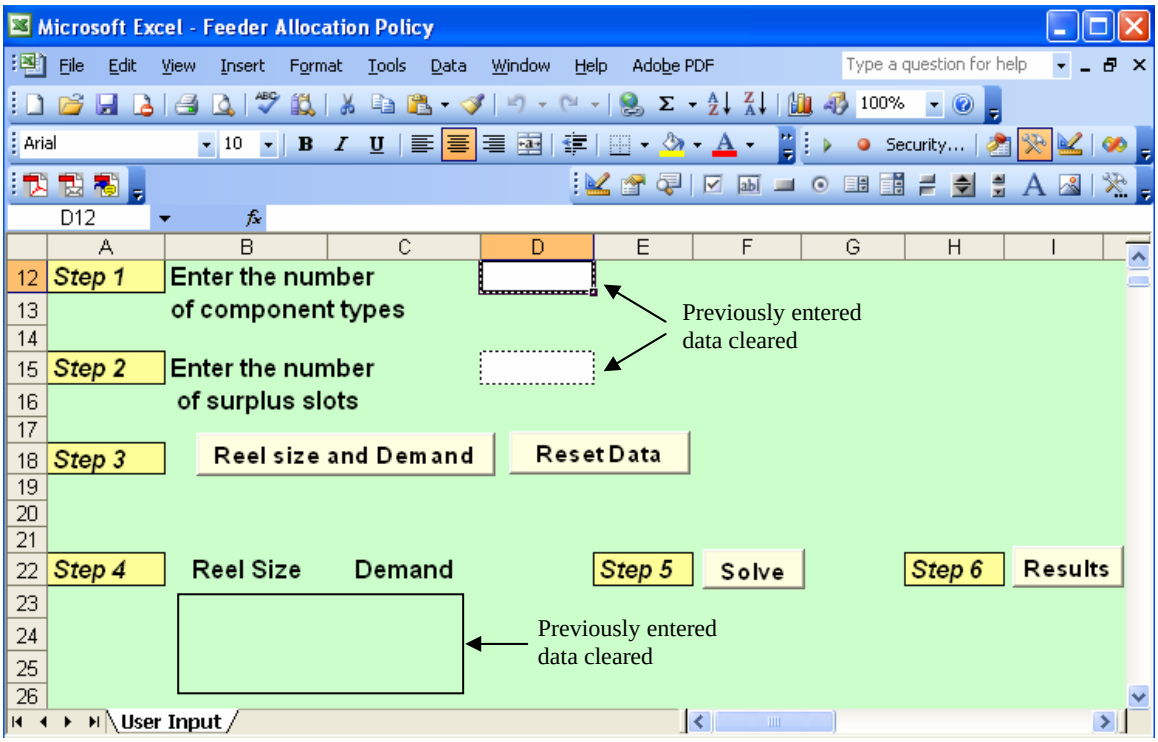

Figure 4.14: Example (2) of resetting data

Another design related issue is that the end users should also be restricted from making any modifications to any of the worksheets. Most importantly the users should be prevented from adding or deleting rows or columns. This can be done by password protecting the worksheets.

4.2.3 Building the model

As the choice of the model and the decisions pertaining to the model like the inputs and outputs have already been identified in Step 1 of the procedure, building the model is a relatively easy and a simple task.

The first step in building the model, is formulating the model. For the reel allocation problem, the model has been formulated as a dynamic programming model which has been explained in chapter 3. After formulation, the second step involves

coding the model. Coding the model can be done using VBA. This is one of the advantages of using Excel. Almost all of the functions that Excel provides are also available in the VBA. The cells in the Excel worksheet can be referenced in the Visual Basic program and assigned to any variable by using the following syntax.

Variable name = Application.SheetName.Cells (a, b)

Or

Variable name = Application.SheetName.Range (“Range Values”)

where a and b are the cells coordinates. By referencing cells in the manner shown above, the integration of the worksheets and VBA is achieved. Excel also provides the flexibility of calling the Solver by using VBA so that the user would not have to do it manually. Moreover it also prevents the user from making changes to the Solver. The procedure of calling Solver from VBA is shown below.

SolverOk

SetCell:= Objective function cell

MaxMinval:= 1,2 or 3

By changing:= decision variable cells

Solver Add cellRef:= constraint cells

The SolverOk command is used to define the objective function and the decision variables. The SetCell command is used to define objective function and the By changing command is used to define the decision variables. MaxMinval is used to denote whether the objective function is maximized (1), minimized (2) or solved for a particular value (3). The Solver Add cellRef command is used to specify the constraints. The SolverReset

command is used each time before the Solver is called to reset all the previously entered objective function, decision variables and constraints.

While programming, it is advisable to use comments in appropriate places, in case some modification to the model would have to done by a different person in future.

4.2.4 Integration

In the methodology proposed in this thesis, integration refers to two levels of integration. First the VBA must be integrated with the worksheets and then the different worksheets must be integrated together. This is done so that the entire interface works seamlessly.

To begin with input sheet should be integrated with the VBA and then the VBA module should be integrated with the calculation sheet (if a calculation sheet exists) and the result sheets. As described in the previous step a part of this integration can be achieved by means of cell referencing. For achieving full integration, buttons can be used. By full integration, it is meant that the once all the data have been input, by just one click of the button, the interface must accept the input (integrating input sheet with the VBA), process the input (integrating the VBA with the calculation sheet) and finally display the results in the results sheet (integrating the VBA with the results sheet). For example in the user interface considered in this thesis, the functions stated above are achieved by means of the button named “Solve”. Once the user enters all the required data and presses the “Solve” button, the results are automatically displayed in the results page. Usually when the “Solve” button is pressed the screen tends to flicker when

processing the data. This can be avoided by using Excel's screen updating command in the Visual Basic code, which is shown below.

Application.ScreenUpdating = False

In order to make the application user friendly, messages should be displayed to the user at appropriate times. For example when the user forgets to enter data and tries to solve the problem or when a solution is found, messages must be displayed indicating that. This can be done by using message boxes in the Visual Basic code. An example of this is shown below in Figures 4.15 and 4.16.

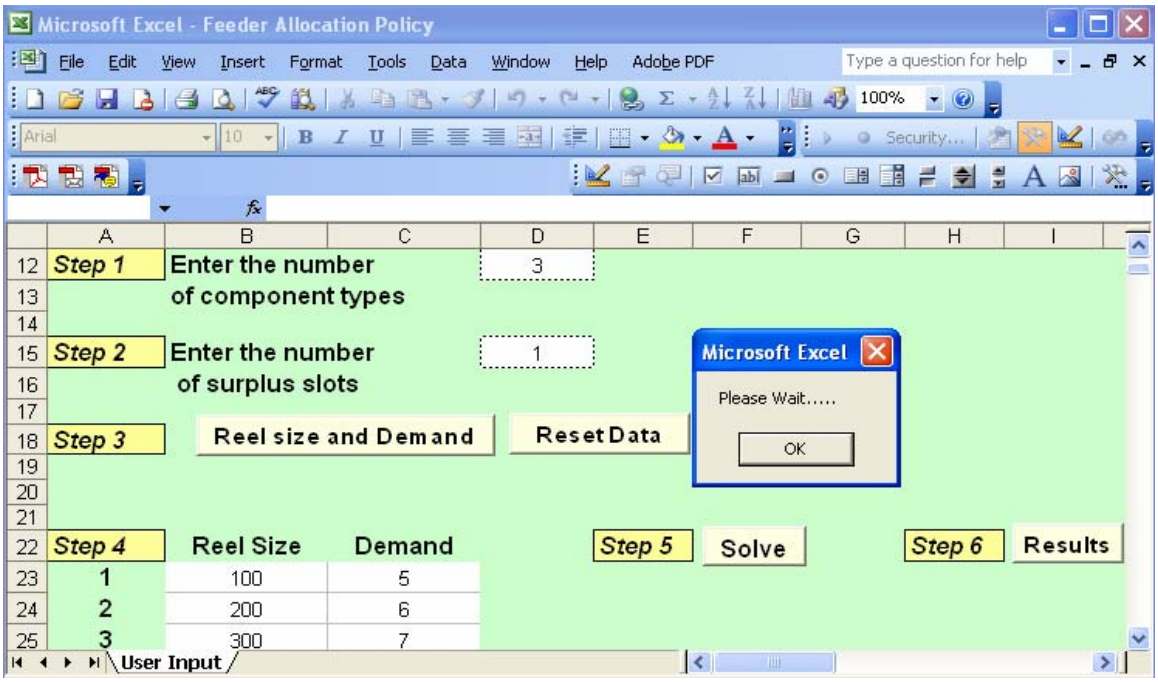

Figure 4.15: Example (1) of messages

When the user clicks on the "Solve" button a message indicating the user to wait is displayed as shown in Figure 4.15. When the user clicks on "OK", another message indicating that the optimal allocation plan is obtained is displayed as shown in Figure 4.16.

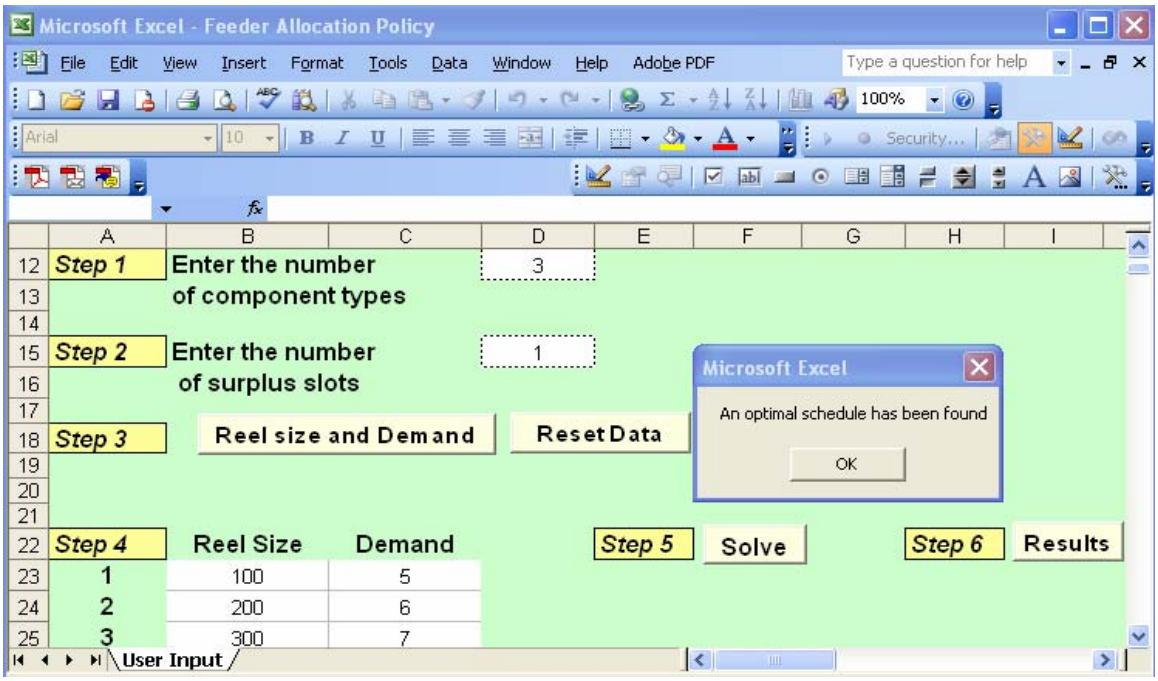

Figure 4.16: Example (2) of messages

The second level of integration deals with the integration of the individual worksheets; the introductory sheet, the input sheet, calculation sheet and the results sheet. At any point in time, it is advisable to make only one worksheet be visible to the user. That is the user can navigate from one sheet to another by the use of buttons. Thus buttons serve to integrate one sheet with another. For example in the interface constructed in this thesis, when Excel is opened each time, only the introductory or the welcome sheet is visible. This is achieved by using the following set of code.

```
Worksheets ("Welcome").Visible = True
    For Each ws In ActiveWorkbook.Worksheets
        If ws.Name <> "Welcome" Then
            ws.Visible = False
        End If
Next ws
```

The name of the introductory sheet is “Welcome”. Each time Excel is opened, the “Welcome” sheet is made visible and all other sheets are invisible. When the button “Allocation Model” is pressed, the “User Input” sheet is made visible and all other sheets are made invisible by the using same procedure described above, within the code of the button. Similarly when the “Results” button is pressed, the “Results” sheet becomes visible and all other sheets becomes invisible. To return back to the input sheet from the results sheet and back button is provided in the results sheet. By this way, just by using buttons all the sheets are integrated. But it is imperative to note that if a calculation sheet is used, it should always remain invisible to the user.

4.2.5 Testing

After all the sheets have been integrated, it is always best to test the working of the model by running a few examples. This would help in identifying the bugs and correcting the problem. In order to check if the model is giving the correct results, a number of problems must be solved using the model constructed and another already established model (if one exists). For example, for the reel allocation problem considered in this thesis, the output of the dynamic programming model has been tested with the output of the IP formulation proposed by Ahmadhi et al [1].

CHAPTER 5

EXPERIMENTATION AND RESULTS

In this chapter the dynamic programming model for which an Excel based interface has been built is tested with a number of problems. The results of the model are tested with the integer programming model proposed by Ahmadhi et al [1]. The test problems have been divided into three categories based on the size of the problem; small, medium and large scale problems. The small problems deal with 10 to 20 slots for the feeder carriage, the medium size problems deal with 21 to 40 slots, and the large scale problems deal with 40 to 60 slots.

5.1 Small scale problems

The input data for the small scale problems is presented below. Five different scenarios (different PCB types called S1, S2, S3, S4, S5) are presented which vary in the number of component types, number of surplus slots, the demand and the reel sizes. The data were generated from a uniform distribution. The parameters for demand values are between 1 and 30 and the parameters for the reel sizes are between 1000 and 6000. Tables 5.1, 5.2 and 5.3 give the details of each PCB type such as the number of components, number of surplus slots, demand and reel sizes.

Table 5.1: Number of components and surplus slots for small scale problem

PCB type	Number of components	Number of surplus slots
S1	10	10
S2	12	8
S3	15	5
S4	18	2
S5	19	1

Table 5.2: Demand for components of each PCB type for small scale problems

	Component type									
PCB Type	1	2	3	4	5	6	7	8	9	10
S1	10	3	15	22	22	24	1	10	21	4
S2	14	2	15	16	17	19	3	12	15	9
S3	16	4	23	12	12	15	10	16	14	15
S4	2	12	16	16	9	7	13	2	8	3
S5	11	9	7	4	2	13	2	14	12	7
	Component type									
PCB Type	11	12	13	14	15	16	17	18	19	
S1	-	-	-	-	-	-	-	-	-	
S2	14	4	-	-	-	-	-	-	-	
S3	22	5	4	16	3	-	-	-	-	
S4	20	1	21	21	3	17	8	18	-	
S5	10	14	11	13	1	13	7	1	10	

Table 5.3: Reel sizes for components of each PCB type for small scale problems

	Component type									
PCB Type	1	2	3	4	5	6	7	8	9	10
S1	2011	4160	2640	3884	3764	4652	4565	2459	2327	2011
S2	1949	2081	5809	4377	5541	5584	4936	5677	2833	1949
S3	3557	2350	4230	5401	4189	5227	4496	3284	3422	3557
S4	4710	3233	2459	1384	4345	5111	4800	4467	2683	4710
S5	3536	3046	4593	3905	4053	2042	2219	3406	4753	3536
	Component type									
PCB Type	11	12	13	14	15	16	17	18	19	
S1	-	-	-	-	-	-	-	-	-	
S2	3419	2419	-	-	-	-	-	-	-	
S3	2140	5165	1312	1423	5284	-	-	-	-	
S4	3636	4709	4968	5416	2927	3752	3942	4652	-	
S5	3324	5278	2394	1664	5853	3228	5974	1540	5682	

The problems were solved using the DP model implemented using VBA. In order to test the accuracy of the DP model, the IP proposed by Ahmadhi et al [1] model was solved using Cplex version 8. All the problems were solved using a Pentium IV, 2GHz, 256 MB RAM computer. Table 5.4 shows the number of boards that can be made by a single allocation of components with and without using extra slots. Table 5.4 also presents the result obtained from the IP model proposed by Ahmadhi et al [1].

Table 5.4: Solution for small scale problems

	Number of boards		
PCB type	Without extra slots	Using extra slots (DP model)	Using extra slots (IP model-Cplex)
S1	117	360	360
S2	138	378	378
S3	67	177	177
S4	153	181	181
S5	128	158	158

From Table 5.4 it can be seen that the DP model gives the same results as the IP model. Table 5.5 shows the component types allocated to the extra slots for each of the PCB types.

Table 5.5: Component types allocated to the extra slots for small scale problems

	Component types	
PCB type	DP model	IP model
S1	3,4,4,5,5,6,6,9,9,9	3,4,4,5,5,6,6,9,9,9
S2	1,3,3,4,5,6,10,11	1,3,3,4,5,6,10,11
S3	1,1,3,11,14	1,1,3,11,14
S4	4,5	4,5
S5	14	14

5.2 Medium scale problems

The input data for the medium scale problems is presented below. All the data were randomly generated from uniform distribution. The parameters for demand values

are between 1 and 30 and the parameters for the reel sizes are between 1000 and 6000. Five different scenarios (PCB types called M1, M2, M3, M4 and M5) are considered. Tables 5.6, 5.7 and 5.8 give the number of components, number of surplus slots, the reel sizes and demand values for each of the PCB types.

Table 5.6: Number of components and surplus slots for medium scale problem

PCB type	Number of components	Number of surplus slots
M1	22	18
M2	25	15
M3	30	10
M4	35	5
M5	38	2

Table 5.7: Demand for components of each PCB type for medium problems

	Component type														
PCB type	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M1	1	2	2	1	3	1	3	3	1	2	3	3	1	2	3
M2	19	10	6	5	4	4	2	2	1	1	1	1	1	6	4
M3	8	2	12	18	17	19	1	8	17	3	5	1	1	4	5
M4	4	1	6	2	9	8	2	7	2	2	1	6	9	1	6
M5	9	4	4	9	1	6	5	1	1	9	8	5	5	6	7
	Component type														
PCB type	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
M1	2	3	1	1	2	5	11	-	-	-	-	-	-	-	-
M2	4	2	2	1	1	1	1	1	1	1	-	-	-	-	-
M3	1	6	7	11	7	8	7	18	9	9	6	19	16	19	5
M4	5	5	2	1	9	4	5	3	4	8	1	1	4	4	4
M5	5	5	6	8	2	2	5	1	6	9	1	7	2	9	9
	Component type														
PCB type	31	32	33	34	35	36	37	38							
M1	-	-	-	-	-	-	-	-							
M2	-	-	-	-	-	-	-	-							
M3	-	-	-	-	-	-	-	-							
M4	2	6	9	7	9	-	-	-							
M5	5	7	9	7	2	6	1	7							

Table 5.8: Reel sizes for components of each PCB type for medium scale problems

	Component type									
PCB type	1	2	3	4	5	6	7	8	9	10
M1	3669	3565	2128	1228	1958	1251	1381	1956	3184	1247
M2	4200	3000	3200	3800	2600	4400	3000	3000	4400	4500
M3	3152	3285	3661	4415	1882	4180	1540	1622	4108	4805
M4	4127	1194	4770	2563	3055	3687	2314	2316	4306	1408
M5	2192	2425	3148	4163	3218	2814	2559	3272	4658	1824
	Component type									
PCB type	11	12	13	14	15	16	17	18	19	20
M1	3566	2615	1598	3365	1133	2028	1633	3737	1302	3182
M2	4800	3900	3200	4300	3900	3600	4900	4800	3800	4400
M3	3203	1695	1719	3466	1810	2885	4746	1694	3897	2231
M4	3702	1562	3126	2911	2407	4505	3039	3194	2713	4438
M5	4641	4387	1144	3132	1647	2898	4054	1958	1854	2164
	Component type									
PCB type	21	22	23	24	25	26	27	28	29	30
M1	3456	4523	-	-	-	-	-	-	-	-
M2	4100	4400	4200	4400	3300	-	-	-	-	-
M3	2146	2401	1371	4057	2804	3375	1556	2351	2887	3706
M4	3722	2583	3586	2651	4759	1960	1473	2511	3906	2471
M5	4765	1802	3862	1090	4424	3645	1479	1997	2838	1760
	Component type									
PCB type	31	32	33	34	35	36	37	38		
M1	-	-	-	-	-	-	-	-		
M2	-	-	-	-	-	-	-	-		
M3	-	-	-	-	-	-	-	-		
M4	3684	3833	2488	3840	1964	-	-	-		
M5	4794	1900	1132	2249	1062	4333	2787	2431		

It should be noted that the data for PCB type M2 were obtained from a high volume electronics manufacturer in Griffin, GA. The data for the rest of the PCB types were generated randomly. Table 5.9 shows the number of boards that can be made by a single allocation of components with and without using extra slots. Table 5.9 also presents the result obtained from the IP model proposed by Ahmadhi et al [1].

Table 5.9: Solution for medium scale problems

	Number of boards		
PCB type	Without extra slots	Using extra slots (DP model)	Using extra slots (IP model-Cplex)
M1	377	1233	1233
M2	221	1105	1105
M3	76	228	228
M4	218	347	347
M5	125	195	195

Table 5.10 shows the component types allocated to the extra slots for each of the PCB types.

Table 5.10: Component types allocated to the extra slots for medium scale problems

	Component types	
PCB type	DP model	IP model
M1	3,4,5,7,7,8,10,11,12,15,15, 15,16,17,17,21,22,22	3,4,5,7,7,8,10,11,12,15,15, 15,16,17,17,21,22,22
M2	1,1,1,1,2,2,2,3,3, 4,5,6,14,15,16	1,1,1,1,2,2,2,3,3, 4,5,6,14,15,16
M3	5,5,6,8,23,23,27,27,28,29	5,5,6,8,23,23,27,27,28,29
M4	5,8,12,33,35	5,8,12,33,35
M5	24,33	24,33

5.3 Large scale problems

The input data for the large scale problems is presented below in Tables 5.11, 5.12 and 5.13. All the data were randomly generated from a uniform distribution. The parameters for demand values are between 1 and 30 and the parameters for the reel sizes are between 1000 and 6000. Five different scenarios (PCB types called L1, L2, L3, L4 and L5) are considered.

Table 5.11: Number of components and surplus slots for large scale problem

PCB type	Number of components	Number of surplus slots
L1	40	20
L2	45	15
L3	50	5
L4	55	5
L5	58	2

Table 5.12: Demand for components of each PCB type for large scale problems

	Component type														
PCB type	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L1	5	13	3	7	10	5	1	10	6	1	7	13	10	8	4
L2	16	1	4	9	4	1	16	11	8	13	6	3	16	11	14
L3	18	12	14	17	11	2	7	3	7	11	3	14	19	10	16
L4	5	1	15	8	8	3	2	11	6	15	4	9	5	16	12
L5	5	12	1	6	10	5	13	8	10	3	9	4	9	2	10
	Component type														
PCB type	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
L1	1	9	13	14	7	3	8	10	2	13	11	7	2	2	13
L2	12	13	7	16	6	15	10	16	8	6	12	12	5	10	2
L3	17	13	1	7	9	6	4	2	19	11	16	15	11	3	5
L4	8	11	10	3	7	14	7	3	4	11	12	14	4	11	6
L5	9	9	1	6	10	2	12	5	2	11	14	3	2	5	12
	Component type														
PCB type	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
L1	2	7	7	9	1	1	4	12	7	4	-	-	-	-	-
L2	5	2	8	13	14	3	15	3	7	3	8	17	16	8	6
L3	19	18	19	15	4	15	12	4	8	2	7	17	14	1	17
L4	6	4	10	7	15	9	15	16	11	14	4	5	4	11	2
L5	8	9	7	6	13	2	9	9	9	8	13	9	4	2	2
	Component type														
PCB type	46	47	48	49	50	51	52	53	54	55	56	57	58		
L1	-	-	-	-	-	-	-	-	-	-	-	-	-		
L2	-	-	-	-	-	-	-	-	-	-	-	-	-		
L3	1	14	12	9	16	-	-	-	-	-	-	-	-		
L4	13	3	1	15	12	8	11	8	5	10	-	-	-		
L5	9	11	7	14	10	11	10	4	4	2	6	11	11		

Table 5.13: Reel sizes for components of each PCB type for large scale problems

	Component type									
PCB type	1	2	3	4	5	6	7	8	9	10
L1	1493	2415	2310	1681	2785	5167	3398	2696	4948	3875
L2	4245	5474	1056	1056	4645	4133	4333	1950	4330	4360
L3	3943	5488	2316	4977	2454	1478	4046	2886	2840	2750
L4	1327	4758	4019	5009	5903	3583	1105	4410	1713	1153
L5	2317	2827	4499	1782	1327	3644	2426	1126	1146	2951
	Component type									
PCB type	11	12	13	14	15	16	17	18	19	20
L1	4189	2051	4297	5831	1608	5058	4100	1090	3392	2303
L2	1681	3210	3459	4773	2436	3765	1941	3078	3706	4108
L3	3617	1997	3099	1006	1329	2672	5854	4737	5138	3289
L4	2271	2649	1675	2555	3240	2125	2402	4014	4631	5578
L5	3245	1331	3891	2586	2896	4219	1190	2374	3767	3032
	Component type									
PCB type	21	22	23	24	25	26	27	28	29	30
L1	3726	5676	1603	4707	1578	3379	5346	1017	5745	4453
L2	5338	3954	4845	3503	1380	5148	2289	1103	4727	3821
L3	1756	2136	2512	2774	2507	5165	3878	3421	3026	1166
L4	4368	5529	4693	1031	2703	1601	2740	4594	4415	2710
L5	2066	2990	2601	1245	1009	4242	4145	2095	1865	3895
	Component type									
PCB type	31	32	33	34	35	36	37	38	39	40
L1	2820	2263	3752	1966	5376	5210	5337	4107	3740	4330
L2	3656	4273	5135	2657	2522	4883	1693	2792	2778	1867
L3	3402	1645	4806	5930	5418	1844	4975	4621	2265	5994
L4	5341	5035	5293	2707	4774	1522	2304	4764	1798	3451
L5	1187	2976	3180	4151	3605	2795	4604	2927	1043	4444
	Component type									
PCB type	41	42	43	44	45	46	47	48	49	50
L1	-	-	-	-	-	-	-	-	-	-
L2	3276	3071	3974	5818	2877	-	-	-	-	-
L3	3113	4475	1827	4167	5329	5669	5545	3613	2658	4273
L4	4744	1049	2256	3758	4925	3776	4548	4617	1693	2788
L5	4000	1040	1847	4486	4861	2436	4885	4205	3687	2580
	Component type									
PCB type	51	52	53	54	55	56	57	58		
L1	-	-	-	-	-	-	-	-		
L2	-	-	-	-	-	-	-	-		
L3	-	-	-	-	-	-	-	-		
L4	2504	1950	2542	2331	5694	-	-	-		
L5	2871	2119	4922	2137	1830	2108	1087	1067		

Table 5.14 shows the number of boards that can be made by a single allocation of components with and without using extra slots. Table 5.14 also presents the result obtained from the IP model proposed by Ahmadhi et al [1].

Table 5.14: Solution for large scale problems

	Number of boards		
PCB type	Without extra slots	Using extra slots (DP model)	Using extra slots (IP model)
L1	83	342	342
L2	112	248	248
L3	83	166	166
L4	76	159	159
L5	91	98	98

Table 5.15 shows the component types allocated to the extra slots for each of the PCB types.

Table 5.15: Component types allocated to the extra slots for large scale problems

	Component types	
PCB type	DP model	IP model
L1	1,2,4,5,8,12,12,18,18,18,18, 19,20,23,23,25,25,26,32,34	1,2,4,5,8,12,12,18,18,18,18, 19,20,23,23,25,25,26,32,34
L2	4,4,8,13,15,17,19,25,27,28, 34,35,37,37,42	4,4,8,13,15,17,19,25,27,28, 34,35,37,37,42
L3	3,12,13,14,15,16,24,32,36,43	3,12,13,14,15,16,24,32,36,43
L4	10,10,26,37,49	10,10,26,37,49
L5	25,58	25,58

5.4 Analysis of results

The results obtained from the dynamic programming model for all the problem scenarios are same as the results obtained from the IP model proposed by Ahmadhi et al [1]. That is the DP model gives optimal solutions for the feeder allocation problem. It can also be seen from the results obtained, that using the surplus slots and assigning the right component types to the surplus slots, increases the number of boards that can be made by

a single allocation of components to the feeder carriage and thus replenishment is delayed as much as possible. That is for a given batch size, using the allocation plan obtained from the DP model would result in lesser number of replenishments. In order to illustrate the reduction in the number of replenishments, an experiment was conducted with the medium scale problem. Let us assume the following batch sizes shown in Table 5.16 for the different PCB types of the medium scale problems.

Table 5.16: Batch size for medium scale PCB types

PCB type	Batch Size
M1	3000
M2	5000
M3	5000
M4	6000
M5	6000

Table 5.16 indicates that 3000 boards of PCB type M1 have to be manufactured, 5000 boards of PCB type M2 and so on. The allocation plan obtained from the DP model for each PCB type is used and the number of replenishments that occur during the manufacture of the given batch sizes is shown in Table 5.17.

Table 5.17: Number of replenishments for medium scale problems

	Number of replenishments		
PCB type	Without using extra slots	Using extra slots	% Reduction in replenishments
M1	59	27	54.2
M2	70	31	55.7
M3	499	338	32.6
M4	307	257	16.2
M5	511	473	7.4

It can be seen from Table 5.17 that maximizing the number of boards that can be made by a single allocation of components decreases the number of replenishments. The percentage reduction in number of replenishments is also shown in Table 5.17.

This reduction in number of replenishments basically translates into gain in production time.

In order to further test the model, PCB type M2 was tested by varying the number of surplus slots and the number of boards that can be made by a single allocation of components is noted. This data, along with the percentage reduction in replenishments is provided in Table 5.18.

Table 5.18: Number of surplus slots Vs Number of boards

Number of surplus slots	Number of boards	% Reduction in replenishments
0	221	0.00
1	300	14.29
2	442	17.14
3	533	22.86
4	600	27.14
5	650	28.57
6	663	32.86
7	716	35.71
8	760	40.00
9	884	42.86
10	900	44.29
11	900	45.71
12	975	48.57
13	1066	50.00
14	1100	54.29
15	1105	55.71

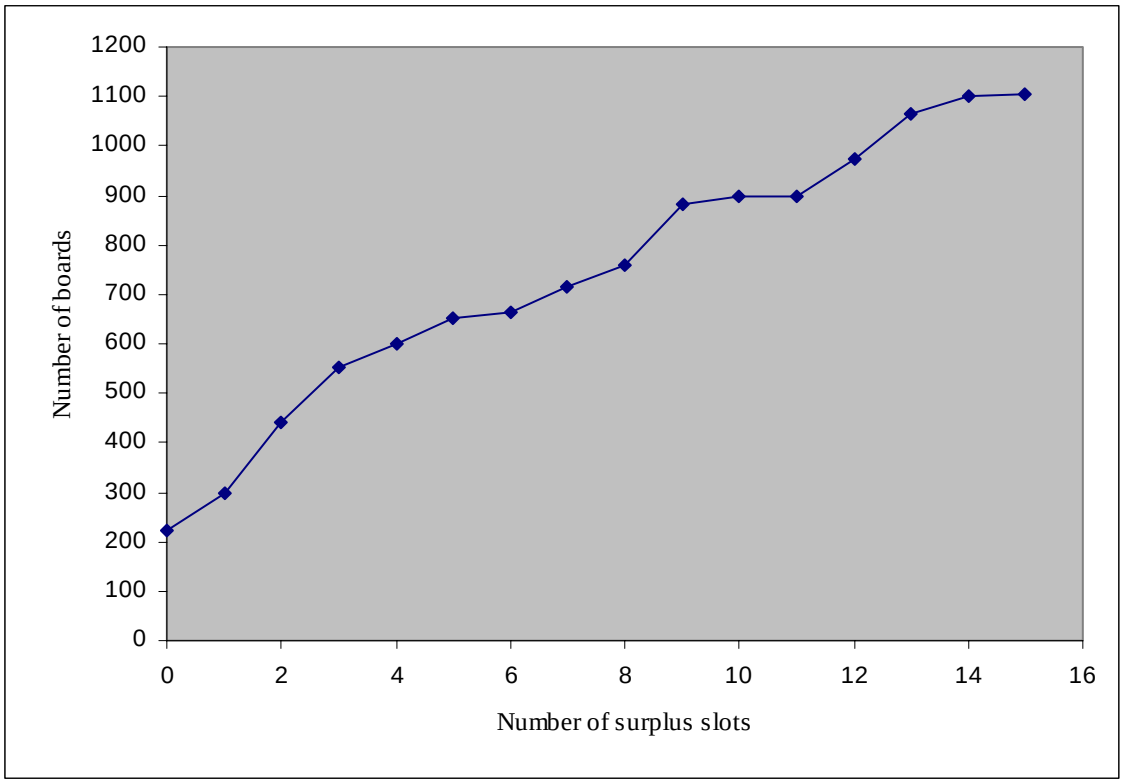

Figure 5.1: Number of boards Vs Number of surplus slots

Figure 5.1 shows the relation between the number of surplus slots used and the number of boards that can be made before the first reel runs out. It can be observed from Figure 5.1 that as the number of surplus slots increases, the number of boards that can be made by a single allocation of components increases, but not at a linear rate. For example the number of boards that can be made by using 2 surplus slots is 442 and the number of boards that can be made by using 3 surplus slots is 533. If the increase in number of boards is linear, when a surplus slot is added each time the number of boards that can be made before the first reel runs out increases by 91. Therefore by using 15 surplus slots 1625 boards can be made. It can be observed from Figure 5.1 that since the increase in number of boards is not linear, the actual number of boards that can be made using 15 surplus slots is 1105.

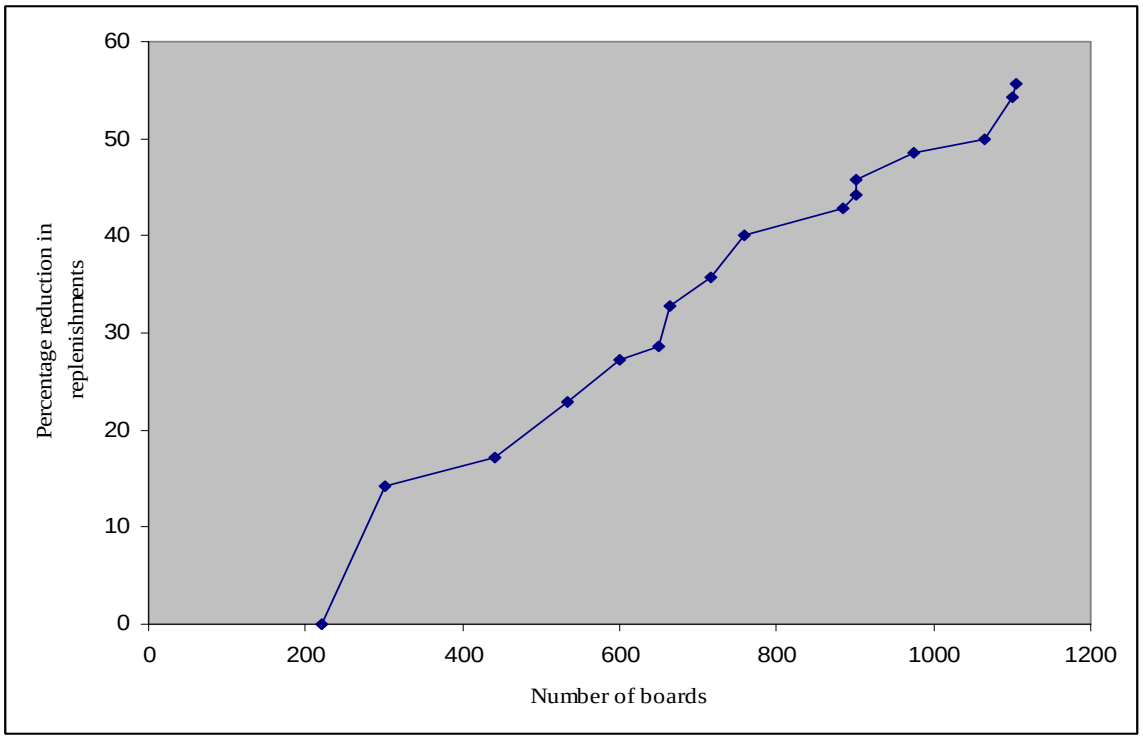

Figure 5.2: Percentage reduction in replenishments Vs Number of boards

Figure 5.2 illustrates the relationship between number of boards that can be made by a single allocation of components and the percentage reduction in number of replenishments. It can be noted that as the number of boards increases, the number of replenishments decreases and the percentage reduction in replenishments increases.

5.4.1 Gain in production time

To estimate the gain in total production time, two variables, T_A and T_B, are defined. The variable T_A is the total time taken for the feeder carriage to move back and forth from the feeder position plus the average time taken for the operator to become available, and the variable T_B is the time taken to replenish a reel. Let η_j represent the number of replenishments without using extra slots when j reels are replenished simultaneously and μ_j the number of replenishments using extra slots when j reels are

replenished simultaneously. Let τ be the production cycle time of the assembly line, P_o the total production time without using extra slots and P_e the total production time using the extra slots. Then P, the relative gain in production time is given by equation (11).

$$P = \frac{P_o - P_e}{P_o} \tag{11}$$

where

$$P_o = \tau \times batchsize + T_A \sum_j \eta_j + T_B \sum_j j \times \eta_j \tag{12}$$

$$P_e = \tau \times batchsize + T_A \sum_j \mu_j + T_B \sum_j j \times \mu_j \tag{13}$$

Table 5.19 gives the relative gain in production time for the medium scale problems. For each of the PCB's it is assumed that T_A=3 minutes, T_B=1 minute, and τ=30 seconds. It should be noted that the relative gain in production time is dependent on the values chosen for the parameters τ, T_A *and* T_B.

Table 5.19: Relative gain in production time

PCB type	P_o *(secs)*	P_e *(secs)*	Relative gain in production time (P)
M1	104220	96480	7.4%
M2	168240	158220	6%
M3	270900	231660	14.5%
M4	254160	241860	4.8%
M5	303720	294420	3.1%

5.5 Solution times

The solution times for all the problems, solved using the DP model and IP model (proposed by Ahmadhi et al [1]) which was implemented using Cplex, were less than a second.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

The optimization of turret head placement machines consists of three sub problems; determining component placement sequence, assigning component types to the feeder slots and determining component retrieval. In this thesis the second sub problem is considered. Another objective of this thesis is to develop a methodology of solving optimization problems using spreadsheets. Spreadsheet based optimization has been illustrated by considering the feeder allocation problem.

As stated earlier, the first objective deals with the feeder allocation problem of a turret head placement machine. The objective of the model is to determine the number of reels of each component type to be used so that replenishment is delayed as much as possible. The model was formulated as a dynamic programming model.

The DP model was coded using VBA provided by Excel. For illustrating the usage of the Solver add-in provided by Excel, a user interface was also built for the integer programming model proposed by Ahmadhi et al [1]. It should be noted that this was done only for illustration purposes, and is not the objective of this thesis. The disadvantage of using the standard version of Excel solver is that it cannot solve

problems that have more that 200 decision variables. Therefore for the feeder allocation problem, it can solve problems up to 199 component types.

In order to test the DP model, a number of experiments were carried out. The model was tested with three different scale of problems; small scale, medium scale and large scale. The results obtained from the DP model were compared to the solutions obtained from the IP model which was solved using Cplex. The solutions obtained from the dynamic programming model were same as those obtained from the integer programming model.

It was also shown that maximizing the number of boards that can be made by a single allocation of components to the feeder slots, delays replenishment as much as possible, leading to a reduction in the number of replenishments. This was illustrated by carrying out experiments on the medium scale problems. For example for the PCB type M2, the percentage reduction in number of replenishments was 55.71%. An important assumption made in this thesis is that there is no variation in reel sizes due to supplier allowances even though in reality reel size variations do occur. However the effect on the replenishments will be minimal since a replenishment might be advanced or delayed only be a couple of boards.

The reduction in number of replenishments translates into gains in production time. For example optimal usage of the surplus slots for PCB type M3 resulted in an 14.5% gain in production time. The relationship between the number of surplus slots used and the number of boards that can me made was studied. It was found that as the number of surplus slots used increases, the number of boards that can be made by a single allocation of components, increases. The relationship between number of boards and the

number of replenishments was also studied. As the number of boards increases, the number of replenishments decreases.

The second objective of this thesis is to develop a methodology of solving optimization problems using spreadsheets. Optimization problems should be solved keeping the cost involved and the end users in mind. Spreadsheets are widely used in every industry and are comparatively cheaper than the commercially available optimization packages. Excel is easy to use and can be used to effectively solve optimization problems.

There are five basic steps in Excel based optimization; identifying end user requirements/defining the objectives, designing the worksheets, building the model, connecting the worksheets, and testing. Each of these steps has been discussed in detail.

6.2 Future work.

The work described in the previous chapters can be extended by removing some of the assumptions that were made during the formulation of the dynamic programming model. For example, in this thesis it was assumed that each component type needs only one slot. A new model can be formulated by considering the possibility that component types may require more than one slot. Another assumption made in this thesis is that, there are no variations in the reel sizes due to supplier allowances. Future work can be done by considering the variation in reel sizes implicitly in the model.

BIBLIOGRAPHY

1. Ahmadi, J., S. Grotzinger, D. Johnson, “Component allocation and partitioning for dual delivery placement research”, *Operations Research*, pp 176–191, 1988.
2. Albright, S. C., Winston, A., “Spreadsheet Modeling and Applications: Essentials of Practical Management Science”, Thomson Brooks/Cole, 2005.
3. Ammons, J.C., M. Carlyle, L. Cranmer, G.W. DePuy, K.P. Ellis, L.F. McGinnis, C.A. Tovey, and H. Xu, “Component allocation to balance workload in printed circuit card assembly systems”, *IIE Transactions*, pp 265-275, 1997.
4. Baker, K. R., “Optimization Modeling with spreadsheets”, Thomson Brooks/Cole, 2006.
5. Bard, J. F., Clayton. R. W., Feo. T. A., “Machine setup and component placement in printed circuit board assembly”, *International Journal of Flexible Manufacturing Systems*, vol. 6, pp 5-31, 1994.
6. Crama, Y., Flippo. O. E., Spieksma, F. C. R., “The component retrieval problem in printed circuit board assembly”, *International Journal of Flexible Manufacturing Systems*, vol. 8, 287-312, 1996.
7. Crama, Y., Flippo. O. E., Spieksma, F. C. R., “The assembly of printed circuit boards: A case with multiple machines and multiple board types”, *European Journal of Operations Research*, pp 457 – 472, 1997.

8. Denardo, E. V., “The Science of Decision Making: A Problem-Based Approach Using Excel”, John Wiley & Sons Inc, 2002.
9. DePuy, G.W., J.C. Ammons, and L.F. McGinnis, “Multiple assignment of component types of feeder slots on automated printed circuit card placement machines”, *IEEE transactions on Electronics Packaging Manufacturing*, vol. 23 no. 3, pp 157 – 164, 2000.
10. DePuy, G.W., M.W.P. Savelsbergh, J.C. Ammons, L.F. McGinnis, “An integer programming heuristic for component allocation in printed circuit card assembly systems”, *Journal of Heuristics*, pp 351 – 369, 2001.
11. Ellis, K. P., Kobza. J. E., Vittes. J., “Optimizing the performance of a surface mount placement machine”, *IEEE Transaction on Electronics Packaging Manufacturing*, vol. 24, 2001.
12. Ellis, K. P., Kobza. J. E., Vittes. J., “Development of a placement time estimator function for a turret style surface mount placement machine”, *Robotics and Computer Integrated Manufacturing*, pp 241-254, 2002.
13. Francis, R.L., Horak. T., “A note on reel allocation problem, *IIE Transactions*, vol.26 no. 3, pp 111-114, 1994.
14. Gastel, S. V., “A comparison of SMD placement machine concepts”, White paper - Assembléon BV, 2002.
15. Gronalt, M., Grunow. M., Gunther. H. O., Zeller. H., “A heuristic for component switching on SMT placement machines”, *International Journal of Production Economics*, pp 181-190, 1997.

16. Grunow, M., Gunther. H. O., Fohrenbach, A., “Simulation-based performance analysis and optimization of electronics assembly equipment”, *International Journal of Production Research*, vol. 38, no. 17, pp 4247-4259, 2000.
17. Grunow, M., Günther. H. O., Schleusener. M., “Component allocation for printed circuit assembly using modular placement machines”, *International Journal of Production Research*, vol. 41, no. 6, 1311-1331, 2003.
18. Ho, P., Ji. P., “Component scheduling for chip shooter machines: a hybrid genetic algorithm approach”, *Computers and Operations Research*, no. 30, pp 2175-2189, 2003.
19. Nakahara, H., “PCB output 1998”, *Printed Circuit Fabrication*, 1999.
20. Neammanee, P., Sabah. U. R., “Integrated methodology for board assignment and component allocation in printed circuit board assembly”, *International Journal of Production Research*, vol. 41, no. 5, 919-937, 2003.
21. Ramasamy, G., Valenzuela. J. F., Ramaswamy. H. K., “A feeder replenishment policy for a turret head placement machine”, under review, *International Journal of Production Research*.
22. Valenzuela, J. F., Smith. J. S., Evans. J. L., “Allocating solder paste printing inspection in high volume electronics manufacturing”, *IIE Transactions*. Vol. 36, No. 12, pp. 1171-1181, 2004.
23. Zijm, W., van Harten. A., “Process planning for a modular component placement system”, *International Journal of Production Economics*, vol. 30-31, pp 123-135, 1993.

APPENDIX A

VBA CODE FOR THE DP MODEL

```
Function Ft(t As Integer, Xt As Integer) As Integer
    Dim Max As Integer
    Dim Yt As Integer
     If Fvalue(t, Xt) = -1 Then
       If t = T Then
         Fvalue(T, Xt) = Rt(T, Xt)
       Else
         Max = -1
          For Yt = 0 To Xt
              Z = Minimum(Rt(t, Xt), Ft(t + 1, Xt - Yt))
              If Z > Max Then
                 Max = Z
                 best(t, Xt) = Yt
              End If
           Next Yt
          Fvalue(t, Xt) = Max
        End If
```

```
        Ft = Fvalue(t, Xt)
      End If
End Function
Function Rt(t As Integer, Yt As Integer) As Integer
   Rt = Int(r(t) * (Yt + 1) / d(t))
End Function
```

The solution is obtained by calling the function Ft(1,S).

APPENDIX B

VBA CODE FOR THE SEQUENTIAL SEARCH APPROACH

```
Private Sub SequentialSearch( )
  For t = 1 To T
    a(t) = Int(r(t) / d(t))
  Next t
  For s = 1 To S
     Sort(a, I)       ‘ Sort in ascending order where I is the array index
     q = I(1)
     Y(q) = Y(q) + 1
     a(q) = Int(r(q) * (Y(q) + 1) / d(q))
  Next s
End Sub
```