A Stochastic Product Priority Optimization Method for Remanufacturing System Based on Genetic Algorithm

Xiong Xiong
University of Texas-Pan American

A Stochastic Product Priority Optimization Method for Remanufacturing System Based on Genetic Algorithm

A Thesis

by

Xiong Xiong

Submitted to the Graduate School of the
University of Texas-Pan American
In partial fulfillment of the requirements for the degree of

Master of Science

August 2010

Major Subject: Manufacturing Engineering

A Stochastic Product Priority Optimization Method for Remanufacturing System Based on Genetic Algorithm

A Thesis
by
Xiong Xiong

August 2010

Abstract

Xiong Xiong, A Stochastic Production Priority Optimization Method for Remanufacturing System Based on Genetic Algorithm. Master of Science (MS), August 2010, 71 pages, 11 tables, 19 figures, and 29 references.

Increasing number of manufacturers are developing remanufacturing facilities to recover end-of-life products for product/component reuse and material recycling while the high uncertainty pattern of returned products complicates the production planning. In this thesis a stochastic production priority optimization method, considering various priority concerns for remanufacturing systems is developed. Priority ranking and matching algorithm is developed to determine the priority rule, using thirteen weighting factors. Queueing models are developed to formulate the objective function, a genetic algorithm is then developed to search optimal solution under different business configurations. Result of this research will provide insights to priority assignment mechanism, which in turn provides support to manufacturers in decision-making in production planning thus improving the performance of remanufacturing systems. Key words: Remanufacturing, Priority, Optimization, Genetic Algorithm, Production planning.

Dedication

This work is dedicated to my parents and family members, who always love me and support me along the journey.

Acknowledgements

I would like to appreciate my advisor, Dr. Jianzhi Li, for his guidance, inspiration.encouragement and invaluable suggestions during my study and research at UTPA. The knowledge and experience accumulated through him in the past two and half years will continue to serve me throughout my study and career.

I would like to extend my thank and appreciation to professors of Manufacturing Engineering Department including Dr. Miguel A. Gonzalez, Dr. John Lloyd, Dr. Douglas Timmer and Mrs. Letty Ocanas for their help throughout out my study at UTPA.

I am ever thankful to my parents who have been a constant source of encouragement throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter I

Introduction

As a strategy to extend product life cycle and reduce carbon foot-print, many companies have begun to set up remanufacturing lines or employed other companies to provide remanufacturing service for those products returned by customers for either warranty repair, upgrades or products that reach the end of their life cycle. Practices of remanufacturing have also shown to be able to build up mutually beneficial relationships between manufacturers and customers, increases customer loyalty and lower manufacturing cost, helps customers reduce operation costs and improve efficiency. Through remanufacturing, the product is first dissembled and then inspected; old and damaged parts are replaced with new ones. Unrecoverable products or cost inefficient products are collected to recapture any usable parts or recyclable materials they may contain.

The efficiency of the remanufacturing is predominantly impaired due to the complicated, dynamic and stochastic character of the product return flow. This reverse flow of material is typically composed of products with different arrival rate, varied types, uncertain condition, quantity, source and destination of return. The remanufacturing industry is thus experiencing unprecedented problems and cost in building efficient remanufacturing planning strategies. On one hand, the return time and quantity can be very different within different products. Even for the same products, the return time and quantity is usually random. Some of the remanufacturing operations such as disposition of materials, cannot be determined until products are broken down in subsystem by remanufacturing (disassembly shop) and have been fully inspected and evaluated. Even for an identified reusable return product, it requires uncertain procedure and stochastic leading time in remanufacturing subjected to its varying conditions. A well-planned production planning and control system will be very important to coordinate the remanufacturing processes, thus keeping a smooth flow and making maximum profit from it.

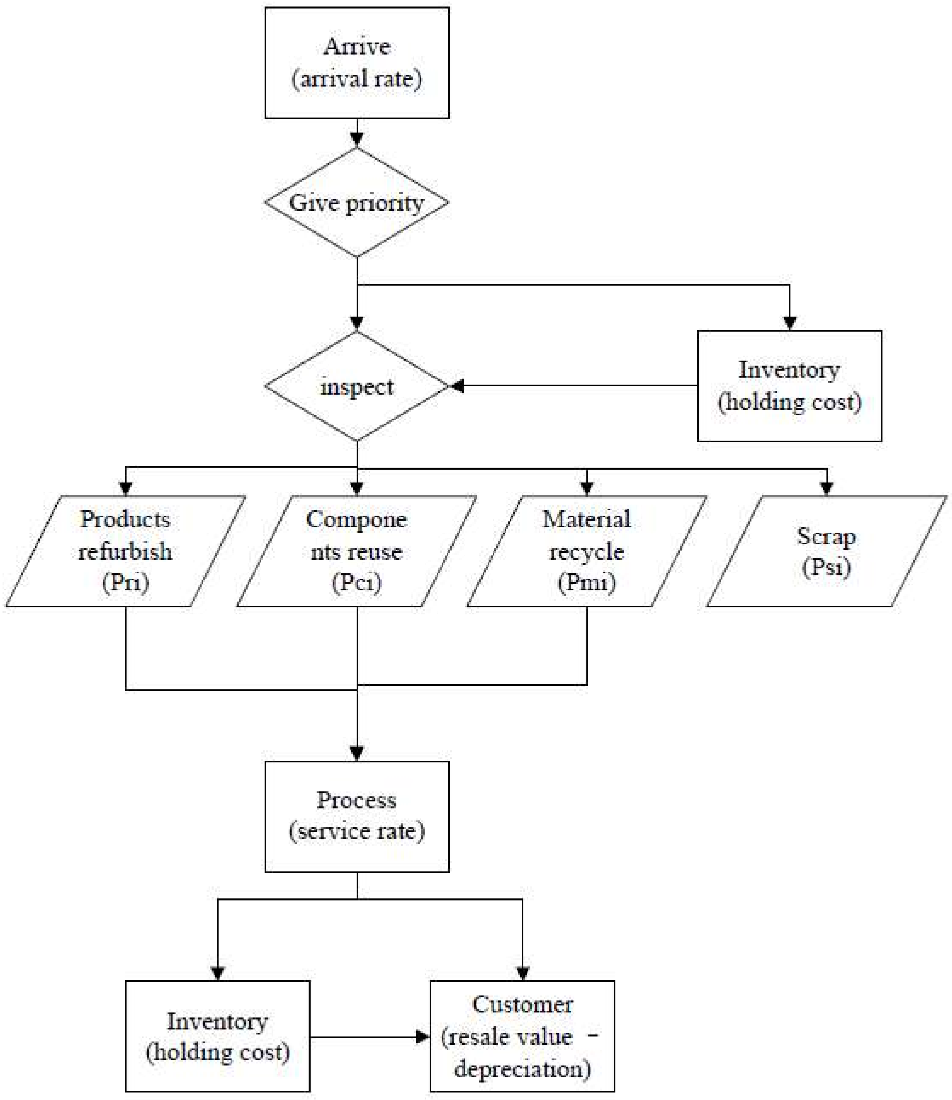


Figure 1: Flowchart of Remanufacturing Process

Remanufacturing originated from military industry and is used to guarantee that, while short of material, the manufacturing will not be interrupted. Now it has been expanded to other fields, especially in the automotive and electronic industries, bringing more than $100 billion annual incomes globally. The typical operations in remanufacturing starts with arrival of used and damaged products from customers, which is a typical compound stochastic arrival process. Received products of varied types and conditions will go through different remanufacturing procedures according to the type, condition, quantity, desired disposition approaches, and existing remanufacturing planning strategy. To ensure the efficient manufacturing process, priorities are generally assigned to products according to different concerns. Products of highest priority will be sent to production line and disassembled into parts, cleaned and classified into different groups. Since the quality of remanufactured product depends largely on the quality of old piece itself, products of which original components are not good enough or remaining life is too short to be used for further remanufacturing will be discarded for material recycling. Products can be repaired and function as new products will be refurbished. For products that cannot be repaired as a whole, reusable parts will be disassembled, refurbished or reconditioned. For parts that can't be repaired, material-recycling strategy will be applied. In summary, products with varied conditions will be repaired with different remanufacturing technologies and strategies, based on priority assigned by manufacturers. Refurbished products will enter market once again and products and parts that have to be recycled will be delivered to the downstream recycling facilities.

It has been widely accepted that the priority mechanism of remanufacturing operations plays an important role in improvement of the performance of remanufacturing business. A stream of research efforts have been dedicated on the evaluation of specific priority dispatching rules in remanufacturing (Li, et al. 2008, Guide et al. 1997, 2005, Erwin, (1997)), with an aim to find best practices and determination of better priority mechanism using comparison and experimental evaluation. It is agreed that a good priority mechanism can not only reduce inventory cost, operating cost, but also improve throughput, resale value and total revenue, thus resulting in a better economic performance for remanufacturing operations. Nonetheless, most research focused on evaluation of a predefined priority rule, very little efforts has delivered to research in identification of an optimal priority mechanism for a certain remanufacturing setup.

Aimed to attack on this problem, in this thesis, a method was developed to determine the optimal priority mechanism using genetic algorithm. Different issues that can be considered in priority determination were first identified. These issues are grouped in for categories: arrival quantity, arrival quality, demand, and processing. In arrival quality, the arrival rate of the different types of products are considered as a main issue. Arrival quality issues mainly include whole product refurbish rate, component reuse rate and material recycling rate. In demand, the main concerns involve resale price for refurbished products, components and recycled materials, the due date of certain products and penalty cost for the delayed products. In processing, issues mainly considered include: process rate, value depreciation due to manufacturing lead-time, setup time and setup cost, holding costs. All these issues are weighted using certain weighting factors and summed into a priority index for a certain product. The same weighting factors are used in determination of the priority indices for all other products that are processed by the same remanufacturers. Based on the priority indices, and ranking procedure is used to assign priority to these different product and determine the overall priority rule based on the weighting factors that were assigned in the beginning. The priority rule will then be evaluated by the fitness function based on the overall profit of the remanufacturer. An iterative process will be conducted by changing the weighting factors and the priority assignments using genetic algorithm, so that an optimal priority assignment can be found to maximize the total profit. The optimal weighting factor also be obtained, which can provide insights for remanufacturing business in determination of their own priority mechanism. This is also the expected contribution of the thesis work.

The thesis is structured as follows: this chapter serves as an introduction to the remanufacturing business, the problem description and brief introduction of the methodology to be developed. Chapter 2 provides a review of literature in related areas. In chapter 3, the research problem is discussed with more details. Chapter 4 provides the methodology and procedures that are developed in this thesis work; A case study is then demonstrated in chapter 5 to validate the research work; Finally in chapter 6, conclusion and suggestions for future work is discussed.

Chapter II

Review of Literature

In general, for most remanufacturing facilities, the product lines are shared by the multiple types of products to be returned. Due to uncertainty in quantity and value that can be recovered, a priority rules need to be created to determine what types of products will be processed with a higher priority. Implementing this priority rule in planning and controlling is crucial to reduce remanufacturing costs and improve the productivity of the remanufacturing system. Several publications have addressed this issue. Guide et al. (1997) compared two planning and controlling strategies: disassembly release mechanisms (DRMs) and priority dispatching rules (PDRs) using simulation considering uncertainty and complexity of remanufacturing. The researchers first introduced the two characteristics that impacted the MPC function: probabilistic recovery rates of the parts from the inducted cores and unknown condition of the recovered parts until inspected. A simulation model was created based on observations from different types of manufacturing workshops, which process mainly large mechanical structures such as airframes, jet turbine engines, jet engine components and heavy diesel engines. Two key activities in MPC, disassembly release mechanisms (DRMs) and priority dispatching rules (PDRs) were examined in the simulation. A (4×16) two-factor ANOVA experiment was designed, which is composed of 4 levels of DRM and 16 levels of PDR, while for each level 15 replications were examined. Mean job flow time, mean job tardiness, percent tardy, and root mean square tardiness were used as performance measures. It shall be noted that cost-based performance measure was not considered in this work. It was concluded that that the DRMs do not provide any significant advantage to any performance measure, so the simplest first come first serve method will be a good choice. Among the PDRs, the earliest due time based rules provides good performance on all the performance measures. Another more complicated remanufacturing model was investigated by Guide et al (2005). In this model, two components from different parts are served in a shared facility with limited capacity. The system performance was measured in term of total weighted average sojourn time (TWAST), ompared under first come first serve (FCFS) priority rule and constrained to simple non-preemptive static priority rules. It is concluded that while both products have the same sojourn time weight, the improvement of using the constrained optimal priority rule was not significant comparing to implementing the simple FCFS rule. If the two products have different sojourn time weights, the system performance was better when simply assigning priority to the products with higher sojourn time weight. John J. Kanet et al. (1982) tested the influence of how well the pre-assigned due dates of arriving jobs are met on job shop performance by introducing the concept of operation due date.

Inventory control for product recovery and remanufacturing has been receiving growing attention. Stochastic hybrid manufacturing / remanufacturing system has received more attention nowadays and a lot of literatures based on it can be found. Erwin et. al (2006) presented their study on a joint remanufacturing and manufacturing job shop under the assumption where remanufacturing is an alternative for manufacturing with lower cost and of the same lead time. Three priority policies were tested under numerical study. The push policy is defined as Whenever the stock of returned items reaches Q_r , those items are remanufactured. Whenever the serviceable inventory position (inventory on hand + everything on order) drops to s_m. A batch of size Q_m is manufactured. The simple pull policy is defined as Whenever the serviceable inventory position drops to the common order level s, a batch of size Q_r is remanufactured if enough returned items are available and a batch of size Q_m is manufactured otherwise. The general pull policy uses separate order levels s_m for manufacturing and s_r for remanufacturing". Minimizing total of manufacturing cost, remanufacturing cost, backorder cost and holding cost is considered as the objective while order size and order level formulae is derived using the same way as traditional inventory system without return. A full factorial design is employed by varying six parameters related to demand and return Poisson process and each priority policy is evaluated in the form of relative cost error in each scenario. Optimal long run average costs are obtained under optimal policy parameter settings of decision variables. Although all of these three priorities perform well, the pull policy is significantly better than the push, and the general pull is slightly better than the simply pull. However, the priority of simple pull is recommended in practical use because of its simple implementation and outstanding results. Katsuhiko et al. dealt with a remanufacturing model with decomposition process where unrecoverable products are decomposed and classified into waste and material which will be used in producing. In this remanufacturing system, products are produced until the stock of products reaches the upper limit γ or the stock of parts runs out. Materials are purchased with the lot size ϵ when the stock of materials runs out. Decomposed materials and/or parts are disposed when the stock of materials or the stock of parts reaches the upper limit, α or β , respectively. Two control policies are proposed with stochastic decomposition process. The first one suggests that parts are produced unless the stock of parts reaches the upper limit β or the stock of materials runs out. The second one proposes that parts are produced unless the stock of parts reaches the threshold δ or the stock of materials runs out. A Markov chain model is developed and steady-state flow balance equations of two policies are derived. The performance of two proposed policies is tested numerically in form of the expected total cost under effects of each rate and cost parameters such as: demand rate, recover rate, production rate of product, production rate of part, production cost of product, production cost of part, disposal cost and shortage cost. Generally the expected total cost of the second policy is lower than the first one. Moritz et al. B. Mahadevan et al. (2003) studied push inventory policies in an inventory system with remanufacturing and manufacturing. The remanufacturing production process is controlled by a periodic review while push policy is operated as following: Every R periods, release stochastic quantity Q_r of all returns stockpile into the remanufacturing facility. I_r is defined as the inventory position of the finished goods stockpile, which is serviceable inventory on hand, less backorders, plus any outstanding (manufacturing or remanufacturing) orders. After releasing the remanufacturing batch, if I_r is less than the manufacturing order-up-to level s_m, there is an order enough products, Q_m from the manufacturing facility to bring inventory position up to s_m . Minimizing the total relevant cost (TC), which is the sum of holding costs for remanufacturable and serviceable items and backorder costs, is considered objective by choosing an appropriate manufacturing order-up-to level s_m. Erwin et al. tested pull and push policies under the effects of lead-time duration and lead-time variability on total expected costs in production/inventory systems with remanufacturing. The PUSH-strategy is defined as remanufacturing starts whenever the inventory of remanufacturables contains Q_r used products. In that case, all Q_r products enter the remanufacturing process to be remanufactured. Manufacturing takes place in batches of size Q_m, and starts whenever the serviceable inventory position (serviceable inventory minus backlog plus all products in (re)manufacturing work in process) drops to the level s_m . The PULL-strategy is defined as remanufacturing starts whenever the serviceable inventory position is at or below s_r, and sufficient remanufacturable inventory exists to increase the serviceable inventory position to s_r. Manufacturing starts whenever the serviceable inventory position drops to the level sm. The manufacturing batch size is Q_m. The total expected costs are evaluated by varying lead-time duration and lead-time variability after obtaining time-average on-hand serviceable inventory and the time-average backordering position of both policies. It turned out that the duration of larger manufacturing lead-times have more effects on total expected costs than the duration of remanufacturing lead-times and cost increases more than when given a larger variability in remanufacturing lead-times than to a larger variability in manufacturing lead-times. It is also observed that sometimes an increase in remanufacturing lead-times or variability in manufacturing lead-times may even result in cost decrease. Karl Inderfurth et al. (2001) introduced a simple 4-factor (serviceable inventory position, quantity of manufacturing products, quantity of returned products, inventory control position) control rule of inventory management in a hybrid manufacturing / remanufacturing model. The optimal inventory position control rule is determined with remanufacturing lead time. The model is tested under a relevant cost structure which consists of variable per unit manufacturing costs, linear holding costs for serviceable products, fixed costs per order for manufacturing and linear backorder costs. It is proved that using remanufacturing lead time as a variable to determine optimal inventory position can improve the performance of system on cost reduction aspect. Karl et al. (2004) presented optimal policies for maximum profit in a situation where remanufactured products can be sold as fully function products, however, for a lower price than new manufactured products.

G.P. Kiesmller (2001) found that remanufacturing inventory control problems are not equal for different lead-time relations after extensive literature review, and used two inventory positions for production decisions and remanufacturing decisions separately. The performance of hybrid stochastic manufacturing/remanufacturing systems is examined under several heuristic policies in term of average costs per time unit. S. Sebnem Ahiska et al. (2008) provided an inventory policy model for manufacturing/remanufacturing systems. The application performance of this model is investigated by changing cost parameters and lead times. It is indicated that his two-parameter policy turned out to be optimal when there are no set up costs for manufacturing and remanufacturing. Jianzhi Li et al. (2008) introduced a hybrid cell evaluated genetic algorithm (CEGA) to optimize dedicated remanufacturing system base on priority dispatching rule using simulation.

V. Daniel R. Guide Jr. (2000) provided a report of current remanufacturing development base on a survey of production planning and control activities at remanufacturing firms in United States ranging from automotive to aero space, machinery and office equipment to small items. With the awareness of the lack of integrated and specialized production and control system in remanufacturing, the author designed the survey. A group of industry experts are selected to provide feedback on the questions about general company information, demand management, material management, production planning and control, and miscellaneous information. The author concluded the seven major characteristics of remanufacturing that significantly complicated the production planning and control activities from the data collected from the survey and their research opportunity. V. Daniel R. Guide Jr. et al. presented a comprehensive survey on production planning and control for remanufacturing, the basic problem of production planning and control for remanufacturing is discussed in which each scenario of remanufacturing process is clearly defined and analyzed . The proposed models are categorized into models for independent demand inventory and models for dependent demand inventory. The author indicated that there are many more case studies or surveys needed to be done related to practical industry. The inherent uncertainties in remanufacturing and integrated models for the management of all supply chain activities are two topics that deserve more attention.

Erwin et al. (1997) investigated the economical benefit of planned disposals in remanufacturing and compared the push-disposal policy to the pull-disposal policy in term of the total expected costs. It is concluded that the priority of disposal policy depends on the cost of holding remanufacturable inventory and holding serviceable inventory. The pull-disposal policy is preferred when remanufacturable inventory is valued sufficiently lower than serviceable inventory; otherwise the push policy is preferred.

Due to high degree of uncertainty in returned products condition, returned products could be either disposed or remanufactured with several options, K. Inderfurth et al. (2001) proposed a model for determining the optimal quantity of products to be remanufactured for each option and disposal, and minimize the relevant cost. Erwin van der Laan et al.(1996) compared three inventory control strategies. For a better understanding of the methodology of production planning under uncertainty, J. Mula, R. Poler presented an extensive review of current research on this topic, and classified the literature to two groups of production planning and modeling approach, and finally identified the future research direction base on the literatures. Christos Zikopoulos et al. (2007) proposed an algorithm to get an optimal disassembly planning strategy. Robert Pellerin et al.(2009).

The nature of our problem suggests a study of a priority queue or a priority scheduling or scheduling function. Literature about these topics has been done previously by other researchers. The study of a priority queue can be found with packet arrival (Walraevens, Wittervrongel, and Bruneel, 2007). In this paper they investigate two types of packets. First class packets have priority and are served first over any other kind of packets. Low priority packets are served when there are no high priority packets present. A study of priority scheduling can be found with packet arrival (Walraevens, Steyaert, Bruneel, 2006). In this paper they analyze discrete-time preemptive repeat queue. Packets of high priority have preemptive repeat priority over those of low priority. In here also when there are no high priority packets low priority packets are served and as soon as a high priority packet arrives, the low priority packet will no be served anymore. Finally a study over service scheduling can be found in telecommunications (Choi, Kim, and Lee, 2007). They studied two types of customers with different service requirements. Customers have a scheduling function to differentiate between them. Customers with certain needs are attended to with certain server and other needs are attended with another server.

Chapter III

Problem Statement

Priority Model in Remanufacturing System

Remanufacturing system is different from traditional manufacturing systems in that it is subjected to a greater degree of uncertainty and complexity in terms of the coming material flow. The returned products generally have a high uncertainty in arrival pattern and high variation in product condition and residual value that can be discovered. High uncertainty has also been experienced with respect to quantity, year of model, and quality of the products received by the remanufacturing system. A product received by the remanufacturers may come from a company that updates its computers regularly so a good condition of the product is reasonably expected. It might also come from a family replacing its 10-year-old home computer, thus a poor condition and virtually no residual value are expected. In addition, one remanufacturing line is generally designed to deal with a group of different products. For products with the same product type, though they are similar, they might have to go through different manufacturing processes according to respective conditions after inspection. The consequence of high uncertainty and variation of the return flow is the difficulty associated in production planning and control of the dedicated model, which leads to increased production cost and poor economic performance.

The profit generated from remanufacturing depends on the value captured from products and components recovered, and the costs spent on the remanufacturing process. Certain priority should be place on products that have higher residual value. In the main time, due to product waiting in a queue of products to be processed, waiting time is expected for most products. Because waiting time can increase the production lead time thus incur higher working-in-progress inventory, it plays an important role in remanufacturing system cost reduction as it is highly related to expenses such as holding cost and depreciation. So it is very important to have a well-programmed planning and control system to reduce the waiting time, minimize the costs, and capture as much revenue, hence maximizing profit. This leads to following three production-planning problems that need to be addressed:

1)

Due to substantial number of product types to be processed by a remanufacturing system, sharing one production line with all product types is better than setting up different lines for each type. Therefore, a priority based switch rule has to be developed for production planning to determine how and when to switch between different production types. The priority mechanism is generally based on following concerns: The first concern is the arrival pattern of a certain product type. Higher arrival rate might call for a higher priority in that this may allow the production line dedicate on one product type thus eliminating line interruption by avoiding frequent product switch and setup of the production line. The second concern is the quality of the product received. No mater what type of product received, the quality condition of each batch of product received may be different. Product with higher reuse potential should be processed first so that value can be recaptured as early as possible. The third group of concern is the demand of the product to be process. Products with a higher resale value need to be processed earlier so that revenue can be realized earlier. Product with an earlier due date and higher penalty cost need to be handled earlier. The fourth group of concerns involves process related concerns. This includes setup time and setup cost for a specific type of products, product with shorter setup time should give higher priority to reduce setup cost and time. Depreciation rate of the products or components received is also a concern. Products with the highest depreciation rate should be given first consideration. These concerns have to be integrated together to generate an optimal priority mechanism and optimal production switch rule for different products, thus reducing and total cost and maximizing the total profit.

2)

When dealing with priority assignment mechanism, most researcher focus on a pre-defined priority assignment rules, ignoring the process that map the priority issues to priority determination. In practice, manufacturers will weight on these concerns considering its business configuration and determine suitable priority structure. Thus, a method that can direct link these concerns to final priority assignment is required. A special weighting based mapping method is proposed in this thesis research. All concerns are aggregated by a group of weighting factors that will be applied to the normalize priority concerns. This aggregated value, which is called priority index, is then ranked. Based on the ranking, priority is assigned to each product group.

3)

In order to maximize the profit of a remanufacturing system, these weighting factor need to be optimized. An objective function has to be developed, which models the expected profit as a function of the assignment priority. Queueing models need to be developed in order to calculate the expected waiting time of different product under a specific priority assignment. Waiting time is ten used to calculate queue length, server utilization rate, holding cost, depreciation costs, and throughput of each product during a specific period of time. This information will then be used to compute the total profit.

4)

Due to the complexity of the optimization problem, evolutionary approaches need to be employed. Invented by John Holland in 1975 based on ideas from the science of genetics and the process of natural selection, Genetic algorithm, is used in many areas to find exact or approximate solutions to optimization and search problems. Suitable crossover, mutation and selection approach need to be chosen.

Chapter IV

Methodology

Research framework and Assumptions

In order to investigate the impact of issues considered in this research to the optimal priority mechanism and to the final net profit, a mathematical model of net profit in typical remanufacturing enterprises has to be developed. All uncertainty issues related to remanufacturing process and their effects on the long-term profit should be identified at first. Initial investigations suggest that there are four groups of issues concerning uncertainty of the returned products. Supply of returned products as first issue is of vital importance, and represented in form of arrival rate.

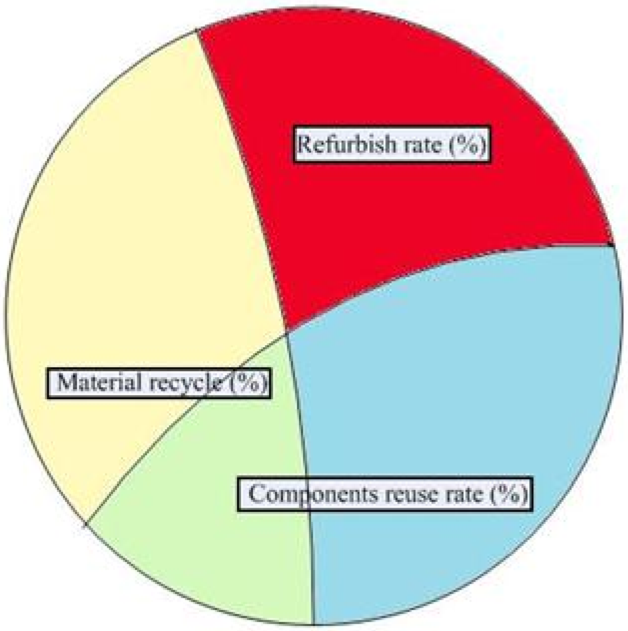


Figure 2: Quality issues of returned products

Studies have shown that the choice of arrival time distribution will not affect performance of remanufacturing workshop (V. Damiel R. Guide Jr et.al 1997). Thus a Poisson distribution for the time between arrivals is generally used in literature and hence assumed in this thesis research. Quality uncertainty is caused by varied usage and condition of returned products. As shown in Figure 2, some of the products are dysfunctional, damaged or just slightly used and can be easily restored and work as newly manufactured products; some cant be fully renewed or serve as a whole product, thus will be dissembled for component reuse or material recycling; a lot of returned products may totally run out of their service life, and will be considered as scrap. Refurbished products, components and material will come back to market with different resale value. A special type of product could be warranty or repair return. Generally, customers would set up a due time when they place the order, and expect the product would be processed within a certain period of time, late penalty may occur when remanufacturer cant accomplish the order before the due time. Holding cost, value depreciation and service rate are three uncertainty factors that directly relate to the remanufacturing process planning. Each issue can be measured in several variables.

The overall research framework is illustrated in figure 2. The variables that affect the optimal priority and net profit provide input to the method. In order to discover the importance of these issues to the production priority and to the profit, the firstly step is to map these variables to a priority index by applying suitable weights to their values. Due to different performance metrics in different variables, values of variables varied from each other. To evaluate each variable on the same level, value of each variable will be normalized. Each variable will be tested while other variable are fixed. Those variables that change in the same direction with profit, in other words the higher variable value the higher profit and the lower variable value the lower profit, are considered as having positive impact on profit, and those variables that change in the opposite direction with profit are considered as having negative impact on profit. By normalization, all the variables will have a range between 0 and 1. For variables that have positive impact on profit, maximum value is recognized as 1 and minimum value is recognized as 0, while variables that have negative impact on profit, maximum value is recognized as 0 and minimum value is recognized as 1.

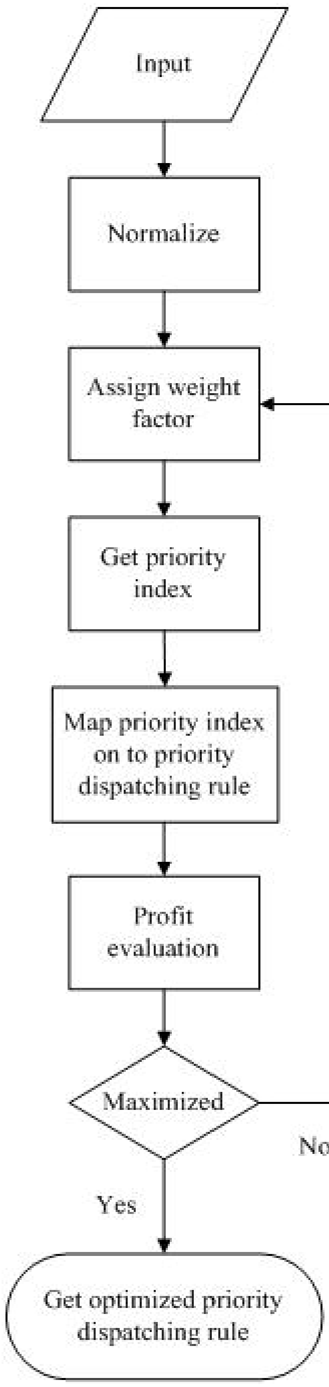

Input
Normalize
Assign weight factor
Get priority index
Map priority index on to priority dispatching rule
Profit evaluation
Maximized
No
Yes
Get optimized priority dispatching rule

Figure 3: Flowchart of Methodology

Initial weighting factors generated by program randomly is assigned to each variable. A variable which is assigned with a higher weight is considered to have more influence on profit. And thus priority index is defined as the sum of each weighting factor times corresponding variable value. Priority index will then be used to rank the priority of n types of product, the higher value of priority index the higher priority of product is.

Once priority is decided, the daily profit can be obtained by the profit equation.

An implementation of the Genetic Algorithm is used to find better weighting factors for all issues, thus resulting in a better performance in profit. The algorithm starts with assigning a fixed number of sets of 15 random integers. The constraint placed on the summation of weights is regulated by making each weight equal to its corresponding random integer divided by the summation of the all 15 random integers, thereby making the sum of all weights equal to 1. After each weight is calculated from each set, products are sorted by priority and total profit is calculated. To know which sets of weights are superior, the sets are sorted by the total profit calculated respectively. For the next generation of sets, the algorithm matches two sets (parents) and exchanges half of their weights to its pair, resulting in two new sets (children) until the number of sets equals its previous generation. In order to minimize similarity between crossovers, the idea of mutation is implemented. There is a fixed probability mutation will occur in a set of weights, which will alter anywhere between one and three weights in that set.

There are two designs in choosing parent sets to create the new generation of sets. The first method of parent choosing takes two percentages relative to the size of the generation, one represents the sets that will be paired and the second being the number of times each set will be paired with the their following sets in the sorted list. Roulette-Wheel selection is then used as the second method of parent choosing, allowing all sets a chance to be paired and decreasing the possibility of remaining at a local maximum. Both methods are used to obtaining a certain percentage of the next generation size, both adding up equal to the size of the previous generation. This variation of the Genetic Algorithm is then used by taking multiple initial populations of sets and allowing them to pair for several generations and taking the maximum found as an approximation to the optimal solution.

Input

The issues that are considered in this research are listed in table 1. They are required in the profit calculation and should be taken into account while assign the priority to arrival return products.

Table 1 Issues in Remanufacturing Process

Issue	Variables	Product Type (r)
Supply	Arrival Rate (unit/day)	λ_r
	Collection cost (dollar/unit)	VCO_r
Quality	Refurbish Rate (%)	PR_r
	Components Reuse Rate (%)	PC_r
	Material Recycle (%)	PM_r
Demand	Due Date (day)	TD_r
	Resale Value (dollar/unit)	VR_r
	Components Reuse Value (dollar/unit)	VC_r
	Material Value (dollar/unit)	VM_r
	Late Penalty (dollar/unit)	VL_r
Remanufacturing Process	Holding Cost (dollar/unit)	VH_r
	Product Depreciation (dollar/unit)	VDP_r
	Component Depreciation (dollar/unit)	VDC_r
	Material Depreciation (dollar/unit)	VDM_r
	Service Rate (unit/day)	μ_r

Issues of r^{th} type of returned product is explained below $(r = 1, 2, \ldots, n)$

λ_r : Arrival rate of product with type r, returned products arrive at a single channel queue according to Poisson process.

VCO_r: Collection cost, which is unit cost for the remanufacturer to get the returned product.

PR_r: Product refurbish rate, which is the percentage of received product r that can be completely repaired and expected to work as newly manufactured products.

PC_r: Components reuse rate, which is percentage of received product r that although cant be fully restored as whole but some of its components still can be reused after dissembling.

PM_r: Material recycle rate, which is percentage of received product r which is badly damaged or totally run out of service life, but the original material still can be recycled and used for other purpose.

TD_r: Due dime, which is the amount of time allowed on the interval between arrival to completion of remanufacturing of a product. Late penalty will be charged when the orders are finished after due date.

VR_r: Whole product resale value, which is the market value of refurbished product r.

VC_r: Components reuse value, which is the value of reusable components collected from product r.

VM_r: Material value, which is the value of recycled materials collected from product r.

VH_r: Holding cost, the handling and inventory cost when the product r stay in the system waiting in a queue or being served.

VDP_r: Depreciation value of product r, which is the decrease in value of products due to obsolescence within the time products stay in the system.

VDC_r: Depreciation value of components from product r within the time components stay in the system.

VDM_r: Depreciation value of material recycled from product r within the time products stay in the system.

μ_r: Service rate, which is the number of product r that a single channel can process every day.

Normalization

Among values of each variable for all returned products, minimums and maximums are picked out, and value of each variable will be normalized as demonstrated in table 2:

Table 2 Normalization of Variable Value

Issue	**Factor**	**Variable of r^{th} Product**	**Minimum Value of Variable**	**Maximum Value of Variable**	**Normalization**
Supply	Arrival Rate (unit/day)	λ_r	$\lambda_{\min}$	$\lambda_{\max}$	$\frac{\lambda_r - \lambda_{\min}}{\lambda_{\max} - \lambda_{\min}}$
	Collection Cost (dollar/unit) / %	VCO_r	$VCO_{\min}$	$VCO_{\max}$	$\frac{VCO_r - VCO_{\min}}{VCO_{\max} - VCO_{\min}}$
Quality	Refurbish Rate (%)	PR_r	$PR_{\min}$	$PR_{\max}$	$\frac{PR_r - PR_{\min}}{PR_{\max} - PR_{\min}}$
	Components Reuse Rate (%)	PC_r	$PC_{\min}$	$PC_{\max}$	$\frac{PC_r - PC_{\min}}{PC_{\max} - PC_{\min}}$
	Material Recycle (%)	PM_r	$PM_{\min}$	$PM_{\max}$	$\frac{PM_r - PM_{\min}}{PM_{\max} - PM_{\min}}$
Demand	Due Time (day)	TD_r	$TD_{\min}$	$TD_{\max}$	$\frac{TD_r - TD_{\min}}{TD_{\max} - TD_{\min}}$
	Resale Value (dollar/unit)	VR_r	$VR_{\min}$	$VR_{\max}$	$\frac{VR_r - VR_{\min}}{VR_{\max} - VR_{\min}}$
	Components Reuse Value (dollar/unit)	VC_r	$VC_{\min}$	$VC_{\max}$	$\frac{VC_r - VC_{\min}}{VC_{\max} - VC_{\min}}$
	Material Value (dollar/unit)	VM_r	$VM_{\min}$	$VM_{\max}$	$\frac{VM_r - VM_{\min}}{VM_{\max} - VM_{\min}}$
	Late Penalty (dollar/unit)	VL_r	$VL_{\min}$	$VL_{\max}$	$\frac{VL_r - VL_{\min}}{VL_{\max} - VL_{\min}}$
Remanufacturing Process	Holding Cost (dollar/day)	VH_r	$VH_{\min}$	$VH_{\max}$	$\frac{VH_r - VH_{\min}}{VH_{\max} - VH_{\min}}$
	Product Depreciation (dollar/day)	VDP_r	$VDP_{\min}$	$VDP_{\max}$	$\frac{VDP_r - VDP_{\min}}{VDP_{\max} - VDP_{\min}}$
	Component Depreciation (dollar/day)	VDC_r	$VDC_{\min}$	$VDC_{\max}$	$\frac{VDC_r - VDC_{\min}}{VDC_{\max} - VDC_{\min}}$
	Material Depreciation (dollar/day)	VDM_r	$VDM_{\min}$	$VDM_{\max}$	$\frac{VDM_r - VDM_{\min}}{VDM_{\max} - VDM_{\min}}$
	Service Rate (unit/day)	μ_r	$\mu_{\min}$	$\mu_{\max}$	$\frac{\mu_r - \mu_{\min}}{\mu_{\max} - \mu_{\min}}$

Assign Weighting Factors

10,000 sets of weighting factors will be generated at first as below:

$$\begin{cases} \overrightarrow{W_1} & = & (w_1^{(1)}, w_2^{(1)}, \ldots, w_{15}^{(1)}) \\ \overrightarrow{W_2} & = & (w_1^{(2)}, w_2^{(2)}, \ldots, w_{15}^{(2)}) \\ . & & \\ . & & \\ . & & \\ \overrightarrow{W_{10{,}000}} & = & (w_1^{(10{,}000)}, w_2^{(10{,}000)}, \ldots, w_{15}^{(10{,}000)}) \end{cases} \tag{1}$$

Where $\sum_{j=1}^{15} w_j = 1$

Weighting factors will be adjusted and reassigned using Genetic Algorithm if necessary. Table 3 summarizes the weighting factors associated with each issue.

Table 3 Weighting Factor for each Variable

Issue	**Variables**	**Product Type (r)**	**Weighting Factor**
Supply	Arrival Rate (unit/day)	λ_r	w_1
	Collection Cost (dollar/day)	VCO_r	w_2
Quality	Refurbish Rate (%)	PR_r	w_3
	Components Reuse Rate (%)	PC_r	w_4
	Material Recycle (%)	PM_r	w_5
Demand	Due Time (day)	TD_r	w_6
	Resale Value (dollar/unit)	VR_r	w_7
	Components Reuse Value (dollar/unit)	VC_r	w_8
	Material Value (dollar/unit)	VM_r	w_9
	Late Penalty (dollar/day)	VL_r	w_{10}
Remanufacturing Process	Holding Cost (dollar/day)	VH_r	w_{11}
	Product Depreciation (dollar/day)	VDP_r	w_{12}
	Component Depreciation (dollar/day)	VDC_r	w_{13}
	Material Depreciation (dollar/day)	VDM_r	w_{14}
	Service Rate	μ_r	w_{15}

Priority Index Calculation

Priority index is used to determine priority dispatching rule

Weight of j^{th} variable: $w_j \qquad (j = 1, 2, \ldots 15)$

Normalized value of j^{th} variable of r^{th} product: $F_{r,j}$

Priority index of r^{th} product: PI_r can be expressed as:

$$\begin{aligned}
PI_r &= \sum_{k=0}^{15} F_{r,k} \times w_k \\
&= \tfrac{\lambda_r - \lambda_{\min}}{\lambda_{\max} - \lambda_{\min}} w_1 + \tfrac{VCO_r - VCO_{\min}}{VCO_{\max} - VCO_{\min}} w_2 + \tfrac{PR_r - PR_{\min}}{PR_{\max} - PR_{\min}} w_3 + \tfrac{PC_r - PC_{\min}}{PC_{\max} - PC_{\min}} w_4 + \tfrac{PM_r - PM_{\min}}{PM_{\max} - PM_{\min}} w_5 \\
&\quad + \tfrac{TD_r - TD_{\min}}{TD_{\max} - TD_{\min}} w_6 + \tfrac{VR_r - VR_{\min}}{VR_{\max} - VR_{\min}} w_7 + \tfrac{VC_r - VC_{\min}}{VC_{\max} - VC_{\min}} w_8 + \tfrac{VM_r - VM_{\min}}{VM_{\max} - VM_{\min}} w_9 + \tfrac{VL_r - VL_{\min}}{VL_{\max} - VL_{\min}} w_{10} \\
&\quad + \tfrac{VH_r - VH_{\min}}{VH_{\max} - VH_{\min}} w_{11} + \tfrac{VDP_r - VDP_{\min}}{VDP_{\max} - VDP_{\min}} w_{12} + \tfrac{VDC_r - VDC_{\min}}{VDC_{\max} - VDC_{\min}} w_{12} + \tfrac{VDM_r - VDM_{\min}}{VDM_{\max} - VDM_{\min}} w_{12} \\
&\quad + \tfrac{\mu_r - \mu_{\min}}{\mu_{\max} - \mu_{\min}} w_{15}
\end{aligned}$$

Where $\sum_{k=1}^{15} w_k = 1$

Map Priority Index to Product Priority

All of PI_r of returned products are ranked from high to low value, where products with higher value of priority index are given higher priority. After finding and sorting products by priority, the products and their corresponding variables are indexed by their priority. Thus, P_r is used to denote product with r^{th} priority. For other variables, index r is also used to represent the priority level.

Profit Evaluation

Profit is used as a performance measure of the priority dispatching rules, which can be obtained based on arrival rate and average waiting time of different types of products.

There are n types of products ranked from priority 1 to priority n after given priority indeces, products of higher priority will always be severed before lower priority, and a unit that begin service complete its service before another unit is admitted. As product of r^{th}

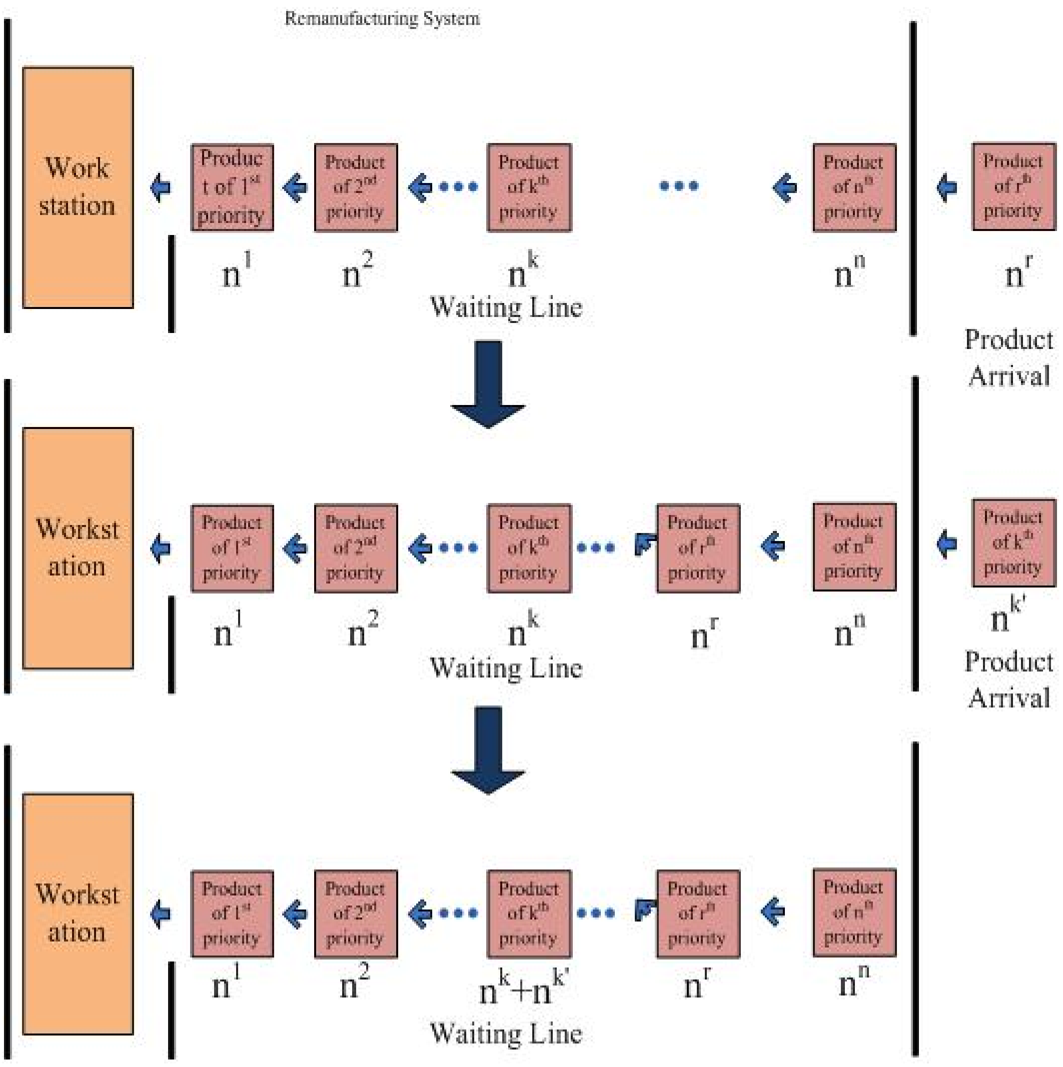


Figure 4: Priority Dispatching Rules

priority arrives at the system with rate λ_r, there are totally $\sum_{1}^{k} n^k$ $(1 \leq k \leq r)$ products of higher priority already in the queue ahead of arriving products. While waiting in the queue more products with higher priority may arrive and be served before r^{th} products. And the process time distribution for the r^{th} priority products is exponential with mean $\dfrac{1}{\mu_r}$.

The expected time of r^{th} product spent in queue is (day/unit)

$$W_q^{(r)} = \frac{\sum_{k=1}^{r} \frac{\rho_k}{\mu_k}}{(1-\sigma_{r-1})(1-\sigma_r)} \qquad \rho_r = \frac{\lambda_r}{\mu_r} \qquad \sigma_r = \sum_{k=1}^{r} \rho_k \qquad \sigma_0 = 0$$

The expected time of r^{th} product spent in system is (day/unit)

$$W^{(r)} = \frac{\sum_{k=1}^{r} \frac{\rho_k}{\mu_k}}{(1-\sigma_{r-1})(1-\sigma_r)} + \frac{1}{\mu_r}$$

Our objective is to maximize annual profit, which is the difference of annual revenue and annual cost. Annual revenue is composed of annual product resale value, annual components reuse value and annual materials recycle value.

$$\begin{aligned}
&\text{Resale Value} &=& \quad \lambda_r PR_r VR_r \\
&\text{Components Reuse Value / day} &=& \quad \lambda_r PC_r VC_r \\
&\text{Materials Recycle Value / day} &=& \quad \lambda_r PM_r VM_r \\
&\text{Revenue / day} &=& \quad \lambda_r PR_r VR_r + \lambda_r PC_r VC_r + \lambda_r PM_r VM_r
\end{aligned}$$

Daily cost consists of collection cost, depreciation, daily holding cost, set up cost and late penalty. Other costs such as labor, tooling, and utility are ignored since there is no relationship between these costs with priority rules. Collection cost is paid by remanufacturer before returned products arrive, which is $\lambda_r VCO_r$. Depreciation happens to all the products while they stay in system, and daily depreciation for r^{th} products is:

$$\lambda_r (PR_r VDP_r + PC_r VDC_r + PM_r VDM_r) W^{(r)}$$

It is assumed that holding cost only occurs when products stay in the queue, daily holding cost for r^{th} product is $VH_r W_q^{(r)}$, and late penalty applied to products that cannot be finished within due time and can be represented as:

$$LP_r \;=\; VL_r \int_{TD_r}^{\infty} \frac{e^{-t/W^{(r)}}}{W^{(r)}} (t - TD_r)\, dt \;=\; VL_r W^{(r)} e^{TD_r/W^{(r)}} \tag{6}$$

Daily net profit of r^{th} product can be expressed as:

$$\begin{aligned}
NP_r \;=\; & \lambda_r (PR_r VR_r + PC_r VC_r + PM_r VM_r - VCO_r) \\
& - \lambda_r (PR_r VDP_r + PC_r VDC_r + PM_r VDM_r) W^{(r)} \\
& - \left(VH_r W_q^{(r)} + VL_r W^{(r)} e^{TD_r/W^{(r)}} \right)
\end{aligned} \tag{7}$$

Total daily profit can be collected from n products as $\sum_{r=1}^{n} NP_r$

Weight Optimization

Genetic Algorithm is used to adjust the initial set of weighting factors until there is maximum profit.
A typical Genetic Algorithm works as following:

[Start] *Generate random population of n chromosomes (suitable solutions for the problem)*

[Fitness] *Evaluate the fitness f(x) of each chromosome x in the population*

[New population] *Create a new population by repeating the following steps until the new population is complete*

[Selection] *Select two parent chromosomes from a population according to their fitness (superior fitness implies the greater chance to be selected)*

[Crossover] *With a crossover probability, cross over the parents to form new offspring (children). If no crossover was performed, offspring is an exact copy of parents.*

[Mutation] *With a mutation probability, mutate new offspring at each locus (position in chromosome).*

[Accepting] *Place new offspring in a new population*

[Replace] *Use new generated population for a further run of algorithm*

[Test] *If the end condition is satisfied, stop and return the best solution in current population*

[Loop] *Go to step 2*

A java program was developed to implement the Genetic Algorithm and to find the optimal weighting factors. The flowchart of the program is illustrated in figure 5.

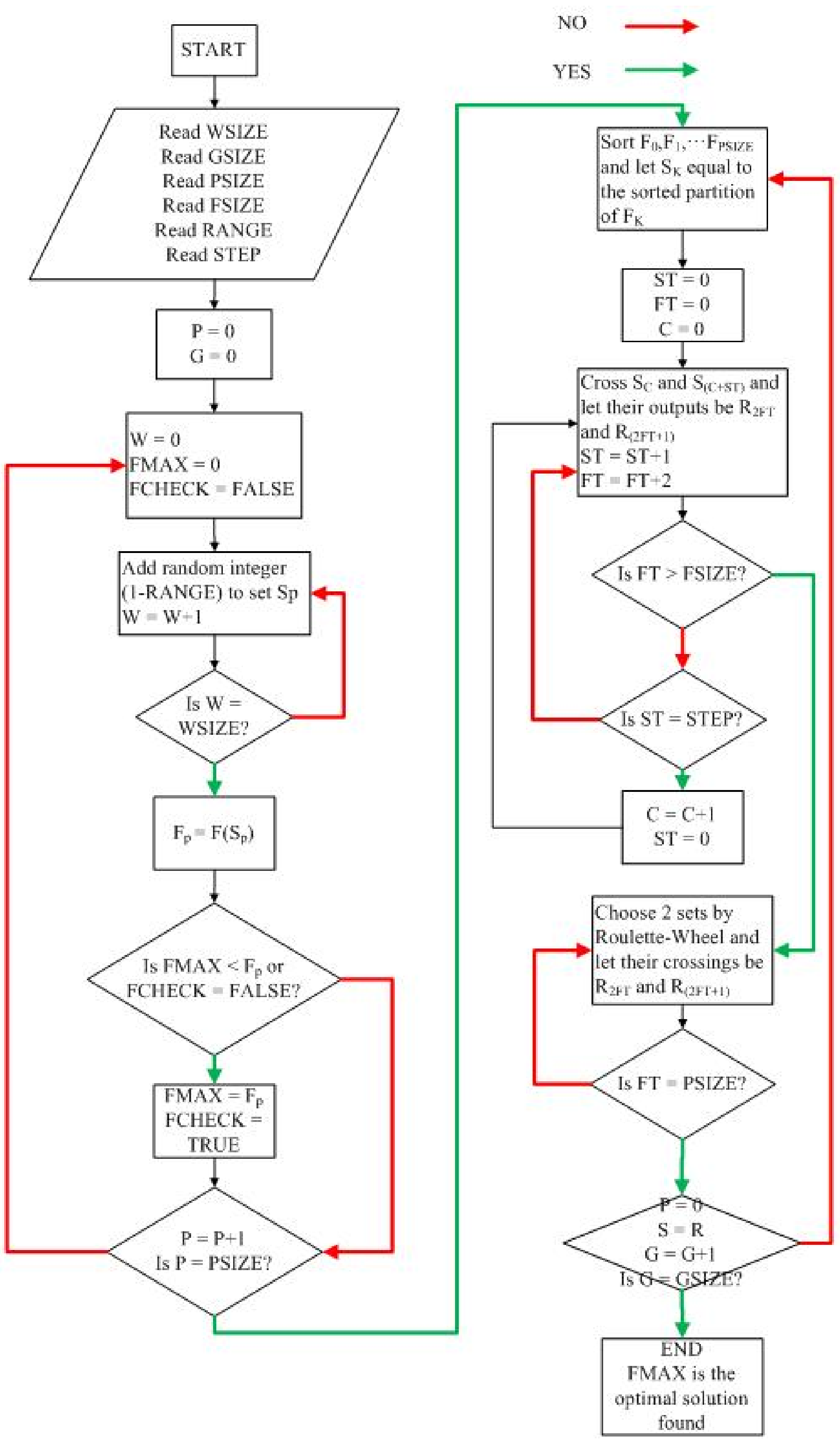

NO
YES
START
Read WSIZE
Read GSIZE
Read PSIZE
Read FSIZE
Read RANGE
Read STEP
P = 0
G = 0
W = 0
FMAX = 0
FCHECK = FALSE
Add random integer (1-RANGE) to set Sp
W = W+1
Is W = WSIZE?
Fp = F(Sp)
Is FMAX < Fp or FCHECK = FALSE?
FMAX = Fp
FCHECK = TRUE
P = P+1
Is P = PSIZE?
Sort F0,F1,⋯FPSIZE and let SK equal to the sorted partition of FK
ST = 0
FT = 0
C = 0
Cross SC and S(C+ST) and let their outputs be R2FT and R(2FT+1)
ST = ST+1
FT = FT+2
Is FT > FSIZE?
Is ST = STEP?
C = C+1
ST = 0
Choose 2 sets by Roulette-Wheel and let their crossings be R2FT and R(2FT+1)
Is FT = PSIZE?
P = 0
S = R
G = G+1
Is G = GSIZE?
END
FMAX is the optimal solution found

Figure 5: Genetic Algorithm Flowchart

The terms and variables used in the program are explained as followed:

$WSIZE$: Number of weights in a set

$GSIZE$: Number of different initial generations used

$PSIZE$: Population size for each generation

$FSIZE$: Fixed number of sets that will crossover

$RANGE$: Range of random integers

$STEP$: Best sets will crossover with $STEP$ consecutive sets until reaching $FSIZE$

P: Number of sets created

G: Number of initial generations created

W: Number of weights currently made for a set

$FMAX$: Best optimal solution found

$FCHECK$: False if no solution has been recorded for $FMAX$, in order to know $FMAX$ has

some value found

S_P: Set P

F_P: Holds total profit using set P

$F(S_P)$: Find total profit using weights from set P

ST:Crossovers for each set and resets to 0 after reaching $STEP$

FT: Number of sets crossover, checked until reaching $FSIZE$

C: Current set used to crossover with next $STEP$ consecutive sets

R_K: Sets used for next population

As shown in Figure 5, the program starts by reading parameters $WSIZE, GSIZE, PSIZE,$ $FSIZE, RANGE, STEP$. Until reaching $PSIZE$ sets, generate $WSIZE$ random weights for each set S_k in the range (1-$RANGE$) and store the calculated total profit for those weights F_k. If F_k is found to be greater than the current $FMAX$, the value of $FMAX$ is exchanged with F_k to keep track of the best profit found. After generating $PSIZE$ sets and computing their total profit, sets are sorted crossovers will replace all $PSIZE$ sets for $GSIZE$ generations.

Two methods of crossovers are used to reproduce new sets for each generation. The first method of crossovers chooses the best sets and crosses them with the next $STEP$ sets until reaching $FSIZE$. Afterwards, Roulette-Wheel is used to choose sets to crossover until reaching $PSIZE$ and finish exchanging all sets to new sets for the new generation. Set replacements for generation to generation is done $GSIZE$ number of times.

Appendix 1 provided the complete code for the program.

Chapter V

Case Study

To validate the methodology developed in this research, a case study is conducted based on a typical remanufacturing system which has been introduced in paper entitled A hybrid simulation optimization method for production planning of dedicated remanufacturing (Jianzhi Li et.al 2009). The facility is a remanufacturing system located in Austin, Texas, which works on recover, reuse and recycle used laptops and desktops for Dell computers. Both returned laptops and desktops are remanufactured in one production line.

As illustrated in figure 6, the products are classified into eight types. This includes desktops and laptop return from different areas with varied models and usage conditions. Product 1 and 2 are computers collected from government and universities, which are normally maintained in good condition and purchased in recent years. A high refurbish rate of 90% is found with returns from this area. Product 3 and 4 are returned from rural area. Due to those computers are generally old models and being used for a long time, 70% of return goes to material recycle, only 10% would be used for refurbish and 20% can be reused as components. Product 5 and 6 come from urban area, as illustrated in Figure 6, 30% of urban returns can be refurbished, 50% can be reused as components, and 20% go to material recycle. Product 7 and 8 are warranty returns, which are newly purchased computers within warranty, returned by customers who are expect to get them back. Warranty returns will be considered as of 100% refurbish rate. Remanufacturer is paid \$50 for repairing each warranty return. \$10 daily late penalty for each unit will be charged if warranty return is not finished within the due time which are 4 days for desktops and 2 days for laptops.

Truck arrivals of return products (except warranty returns) follow a Poisson process with the mean time between arrivals of 4 hour during an 8-hour workday. Laptop and

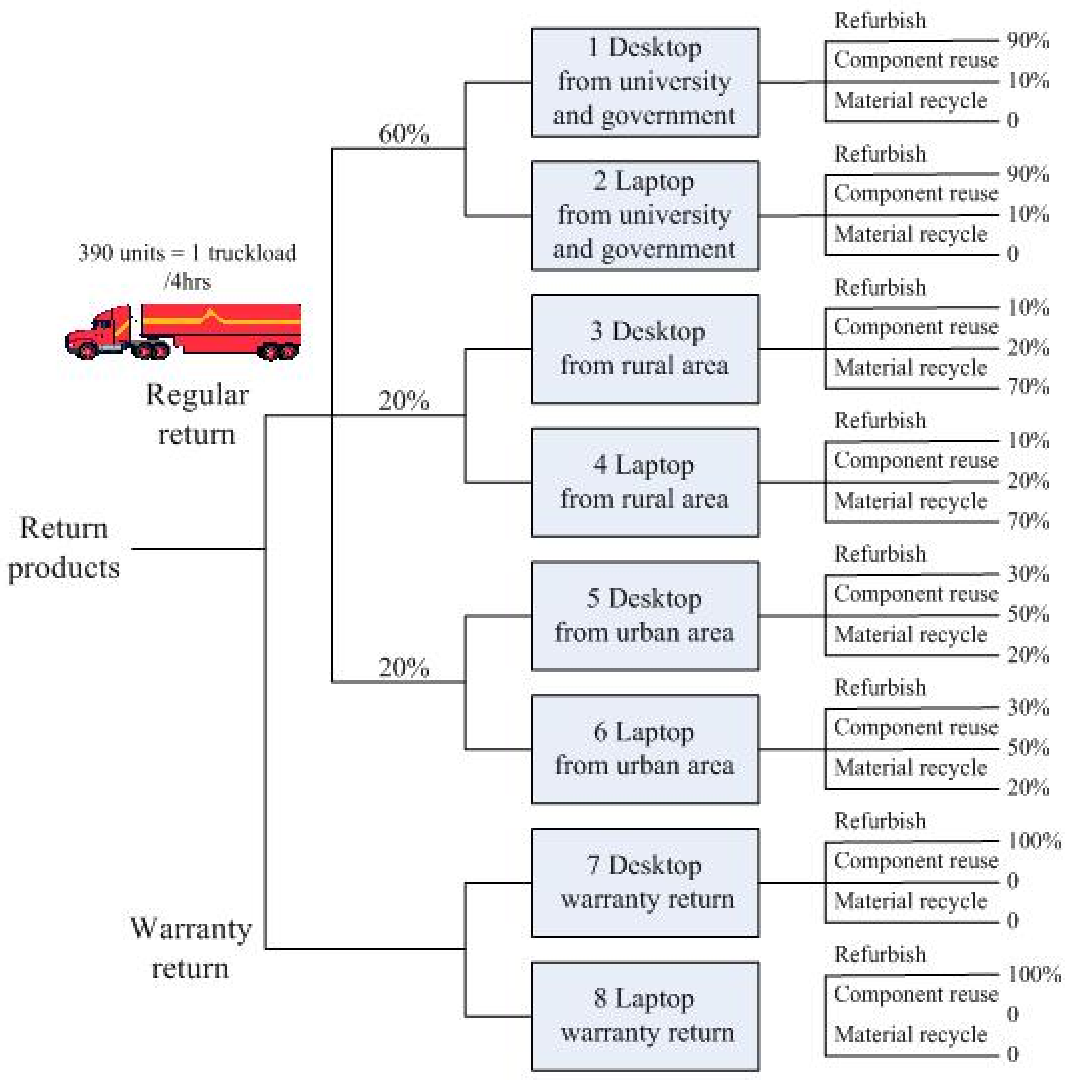


Figure 6: Eight Types of Return Products

desktop come in the same truckload. Normally the proportion of desktops in a truck is as twice as laptops while one truckload is 390 units. Among those returned products 60% are returned by universities and government, 20% come from rural area and the rest 20% come from urban area.

Based on the information, the arrival rate of each product is calculated as follow:

$$\begin{aligned}
\lambda_1 &= \frac{8}{4} \times 390 \times \frac{2}{3} \times 60\% = 312 \text{ units / day} \\
\lambda_2 &= \frac{8}{4} \times 390 \times \frac{1}{3} \times 60\% = 156 \text{ units / day} \\
\lambda_3 &= \frac{8}{4} \times 390 \times \frac{2}{3} \times 20\% = 104 \text{ units / day} \\
\lambda_4 &= \frac{8}{4} \times 390 \times \frac{1}{3} \times 20\% = 52 \text{ units / day} \\
\lambda_5 &= \frac{8}{4} \times 390 \times \frac{2}{3} \times 20\% = 104 \text{ units / day} \\
\lambda_6 &= \frac{8}{4} \times 390 \times \frac{1}{3} \times 20\% = 52 \text{ units / day}
\end{aligned}$$

In average, Dell sends warranty returns with 70 units per day for desktop and 130 units per day for laptops.

As presented in Table 6, collection costs are estimated at \$150 for each desktop and \$250 for each laptop while the selling price of the refurbished desktops and laptops are estimated to be \$250 and \$400 per unit respectively. Components reuse value from each desktop is \$70 and from each laptop is \$100. Material recycle value from each desktop is \$40 and from each laptop is \$30. There is a \$50 revenue from a finished warranty return product.

Holding cost of each product is determined from its value, and according to historical data. Annual holding cost of one product is estimated at 20% of its value, so daily holding cost per unit can be derived as below, all the calculation based on 52 weeks of 5-day weekly schedule:

$$
\begin{aligned}
VH_r &= \frac{20\%(PR_rVR_r + PC_rVC_r + PM_rVM_r}{52 \times 5} \\
VH_1 &= \frac{20\%(90\% \times 250 + 10\% \times 70)}{52 \times 5} &= \$0.1785 \text{ / day} \times \text{ unit} \\
VH_2 &= \frac{20\%(90\% \times 400 + 10\% \times 100)}{52 \times 5} &= \$0.2846 \text{ / day} \times \text{ unit} \\
VH_3 &= \frac{20\%(10\% \times 250 + 20\% \times 70 + 20\% \times 40)}{52 \times 5} &= \$0.0515 \text{ / day} \times \text{ unit} \\
VH_4 &= \frac{20\%(10\% \times 400 + 20\% \times 100 + 20\% \times 30)}{52 \times 5} &= \$0.0623 \text{ / day} \times \text{ unit} \\
VH_5 &= \frac{20\%(30\% \times 250 + 50\% \times 70 + 20\% \times 40)}{52 \times 5} &= \$0.0908 \text{ / day} \times \text{ unit} \\
VH_6 &= \frac{20\%(30\% \times 400 + 50\% \times 100 + 20\% \times 30)}{52 \times 5} &= \$0.1354 \text{ / day} \times \text{ unit}
\end{aligned}
$$

For warranty return holding cost is low and not included in this case.

Deprecation value is also related to products value, annual depreciation is 50% of its value. So daily depreciation for products, components and material of each unit will be:

$$
VDP_r = \frac{50\%\ VR_r}{52 \times 5} \qquad VDC_r = \frac{30\%\ VC_r}{52 \times 5} \qquad VDM_r = 0\%
$$

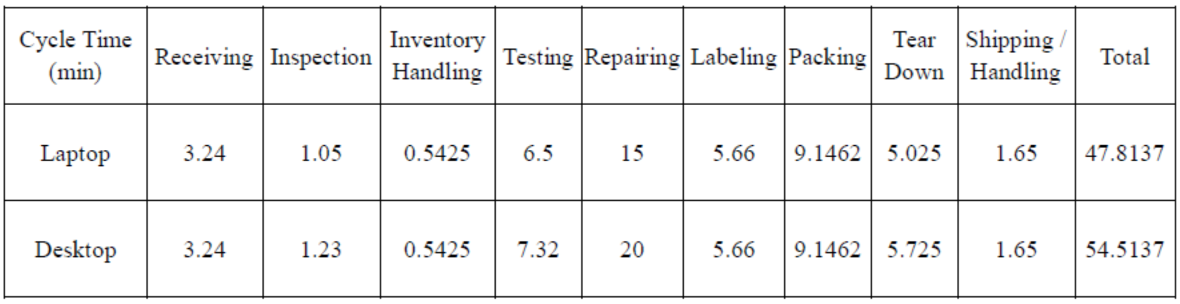
Since depreciations and holding cost in this case totally depend on the value of products, depreciations as VDP_r, VDC_r and VDM_r, holding cost as VH_r will not be considered as variables being assigned weights to. In other words, in this case, only 11 variables as below are assigned weights.

Table 4 Variables are Assigned Weights in this Case

Variables	**Product Type** (r)	**Weighting Factor**
Arrival Rage (unit/day)	λ_r	w_1
Collection Cost (dollar/day)	VCO_r	w_2
Refurbish Rate (%)	PR_r	w_3
Components Reuse Rate (%)	PC_r	w_4
Material Recycle (%)	PM_r	w_5
Due Time (day)	TD_r	w_6
Resale Value (dollar/unit)	VR_r	w_7
Components Reuse Value (dollar/unit)	VC_r	w_8
Material Value (dollar/unit)	VM_r	w_9
Late Penalty (dollar/day)	VL_r	w_{10}
Service Rate	μ_r	w_{11}

Data of cycle time is offered as Table 5 below (Li et al. 2009), it takes 47.8137 minutes for one operator to work on a laptop on average, and it takes 54.5137 minutes for one operator to work on a desktop on average.

Table 5 Cycle Time



Cycle Time (min)	Receiving	Inspection	Inventory Handling	Testing	Repairing	Labeling	Packing	Tear Down	Shipping / Handling	Total
Laptop	3.24	1.05	0.5425	6.5	15	5.66	9.1462	5.025	1.65	47.8137
Desktop	3.24	1.23	0.5425	7.32	20	5.66	9.1462	5.725	1.65	54.5137

There are 53 operators working in two work shifts, so the daily service rate for laptops and desktops are as follows:

$$\text{Laptop} \quad : \quad 8 \times 60 \div 47.8137 \times 53 \times 2 \quad = \quad 1064.13 \text{ units / day}$$

$$\text{Desktop} \quad : \quad 8 \times 60 \div 54.5137 \times 53 \times 2 \quad = \quad 933.34 \text{ units / day}$$

Table 6 Variable Values of Remanufacturing System

Product Type (r)	1	2	3	4	5	6	7	8
	Desktop (Univ.)	Laptop (Univ.)	Desktop (Rural)	Laptop (Rural)	Desktop (Urban)	Laptop (Urban)	Desktop (Warranty)	Laptop (Warranty)
λ_r	312	156	104	52	104	52	70	130
VCO_r	150	250	150	250	150	250	0	0
PR_r	90%	90%	10%	10%	30%	30%	100%	100%
PC_r	10%	10%	20%	20%	50%	50%	0	0
PM_r	0	0	70%	70%	20%	20%	0	0
TD_r	/	/	/	/	/	/	4	2
VR_r	250	400	250	400	250	400	50	50
VC_r	70	100	70	100	70	100	/	/
VM_r	40	30	40	30	40	30	/	/
VL_r	/	/	/	/	/	/	10	10
VH_r	0.1785	0.2846	0.0515	0.0623	0.0908	0.1354	/	/
VDP_r	0.4808	0.7692	0.4808	0.7692	0.4808	0.7692	/	/
VDC_r	0.0808	0.1154	0.0808	0.1154	0.0808	0.1154	/	/
VDM_r	0	0	0	0	0	0	/	/
μ_r	933.34	1064.13	933.34	1064.13	933.34	1064.13	933.34	1064.13

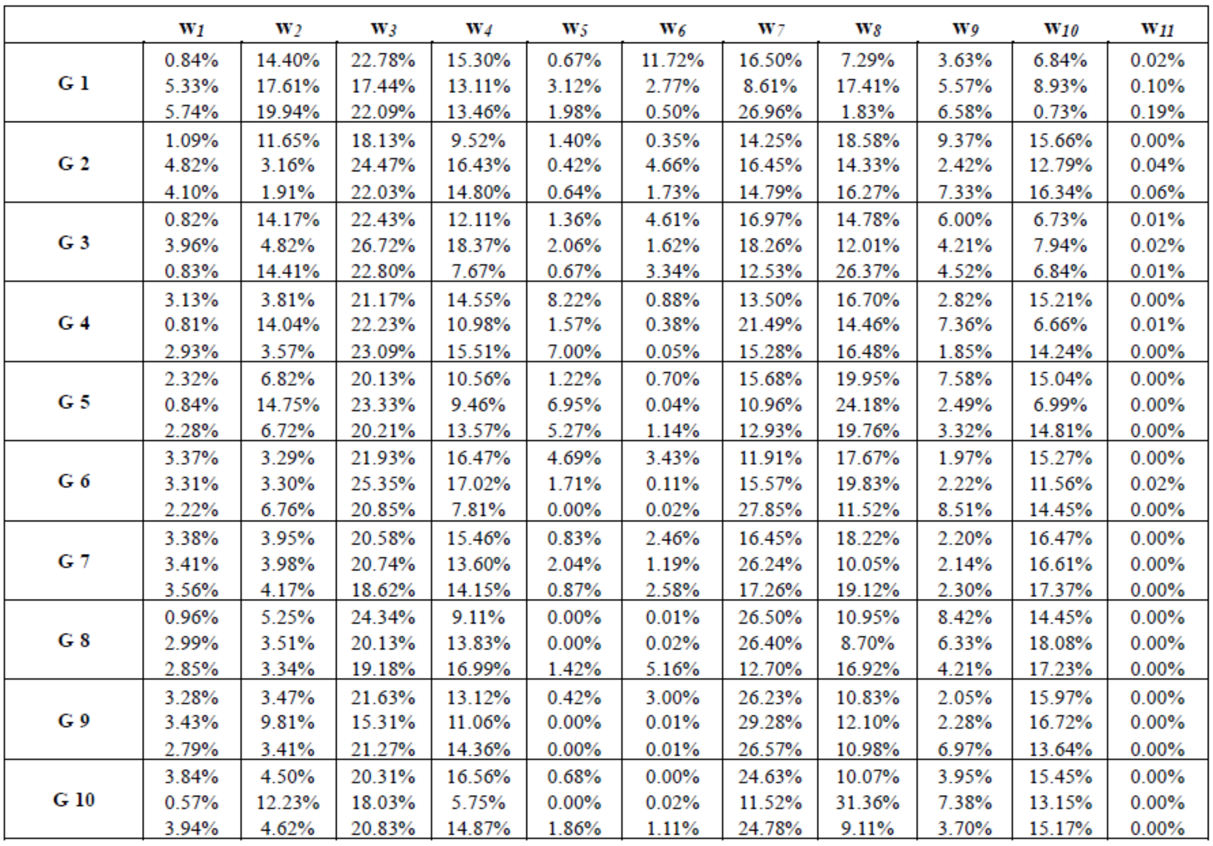
Table 6 summarizes all the data used in this case study. As demonstrated in Chapter 4, to be compared in same scale, all the variable values will be normalized to the range from 0 to 1. Normalized variable value is listed as below in Table 7.

Table 7 Normalized Variables

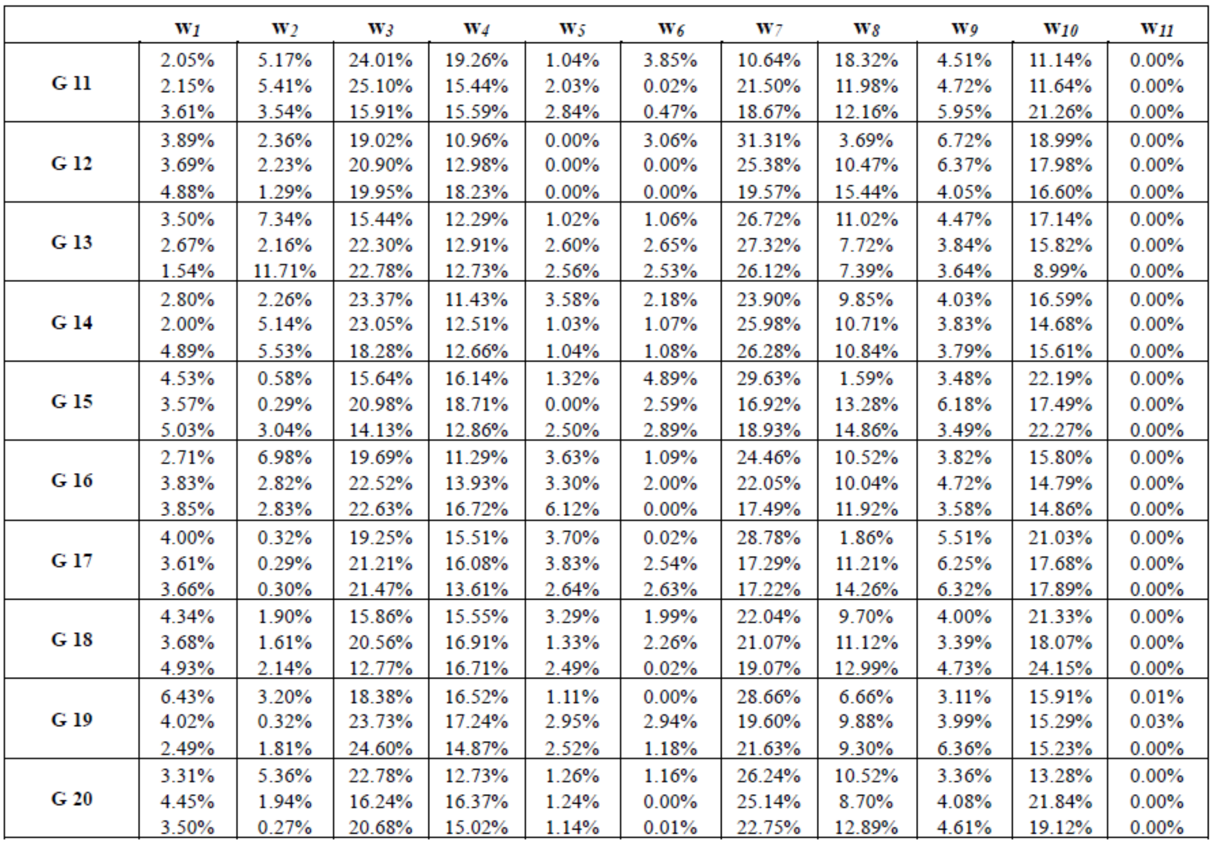
Product Type (r)	1	2	3	4	5	6	7	8
	Desktop (Univ.)	Laptop (Univ.)	Desktop (Rural)	Laptop (Rural)	Desktop (Urban)	Laptop (Urban)	Desktop (Warranty)	Laptop (Warranty)
λ_r	1.00	0.40	0.20	0.00	0.20	0.00	0.07	0.30
VCO_r	0.40	0.00	0.40	0.00	0.40	0.00	1.00	1.00
PR_r	0.89	0.89	0.00	0.00	0.22	0.22	1.00	1.00
PC_r	0.20	0.20	0.40	0.40	1.00	1.00	0.00	0.00
PM_r	0.00	0.00	1.00	1.00	0.29	0.29	0.00	0.00
TD_r	/	/	/	/	/	/	1.00	1.00
VR_r	0.57	1.00	0.57	1.00	0.57	1.00	0.00	0.00
VC_r	0.00	1.00	0.00	1.00	0.00	1.00	/	/
VM_r	1.00	0.00	1.00	0.00	1.00	0.00	/	/
VL_r	/	/	/	/	/	/	1.00	1.00
VH_r	0.46	0.00	1.00	0.95	0.83	0.64	/	/
VDP_r	1.00	0.00	1.00	0.00	1.00	0.00	/	/
VDC_r	1.00	0.00	1.00	0.00	1.00	0.00	/	/
VDM_r	0.00	0.00	0.00	0.00	0.00	0.00	/	/
μ_r	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00

Genetic algorithm is then executed. Three weighting factor sets which generate the top 3 maximum profit are selected from each generation and listed below in Table 8.

Table 8 Weighting Factor Sets from each Generation



	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}
G 1	0.84%	14.40%	22.78%	15.30%	0.67%	11.72%	16.50%	7.29%	3.63%	6.84%	0.02%
	5.33%	17.61%	17.44%	13.11%	3.12%	2.77%	8.61%	17.41%	5.57%	8.93%	0.10%
	5.74%	19.94%	22.09%	13.46%	1.98%	0.50%	26.96%	1.83%	6.58%	0.73%	0.19%
G 2	1.09%	11.65%	18.13%	9.52%	1.40%	0.35%	14.25%	18.58%	9.37%	15.66%	0.00%
	4.82%	3.16%	24.47%	16.43%	0.42%	4.66%	16.45%	14.33%	2.42%	12.79%	0.04%
	4.10%	1.91%	22.03%	14.80%	0.64%	1.73%	14.79%	16.27%	7.33%	16.34%	0.06%
G 3	0.82%	14.17%	22.43%	12.11%	1.36%	4.61%	16.97%	14.78%	6.00%	6.73%	0.01%
	3.96%	4.82%	26.72%	18.37%	2.06%	1.62%	18.26%	12.01%	4.21%	7.94%	0.02%
	0.83%	14.41%	22.80%	7.67%	0.67%	3.34%	12.53%	26.37%	4.52%	6.84%	0.01%
G 4	3.13%	3.81%	21.17%	14.55%	8.22%	0.88%	13.50%	16.70%	2.82%	15.21%	0.00%
	0.81%	14.04%	22.23%	10.98%	1.57%	0.38%	21.49%	14.46%	7.36%	6.66%	0.01%
	2.93%	3.57%	23.09%	15.51%	7.00%	0.05%	15.28%	16.48%	1.85%	14.24%	0.00%
G 5	2.32%	6.82%	20.13%	10.56%	1.22%	0.70%	15.68%	19.95%	7.58%	15.04%	0.00%
	0.84%	14.75%	23.33%	9.46%	6.95%	0.04%	10.96%	24.18%	2.49%	6.99%	0.00%
	2.28%	6.72%	20.21%	13.57%	5.27%	1.14%	12.93%	19.76%	3.32%	14.81%	0.00%
G 6	3.37%	3.29%	21.93%	16.47%	4.69%	3.43%	11.91%	17.67%	1.97%	15.27%	0.00%
	3.31%	3.30%	25.35%	17.02%	1.71%	0.11%	15.57%	19.83%	2.22%	11.56%	0.02%
	2.22%	6.76%	20.85%	7.81%	0.00%	0.02%	27.85%	11.52%	8.51%	14.45%	0.00%
G 7	3.38%	3.95%	20.58%	15.46%	0.83%	2.46%	16.45%	18.22%	2.20%	16.47%	0.00%
	3.41%	3.98%	20.74%	13.60%	2.04%	1.19%	26.24%	10.05%	2.14%	16.61%	0.00%
	3.56%	4.17%	18.62%	14.15%	0.87%	2.58%	17.26%	19.12%	2.30%	17.37%	0.00%
G 8	0.96%	5.25%	24.34%	9.11%	0.00%	0.01%	26.50%	10.95%	8.42%	14.45%	0.00%
	2.99%	3.51%	20.13%	13.83%	0.00%	0.02%	26.40%	8.70%	6.33%	18.08%	0.00%
	2.85%	3.34%	19.18%	16.99%	1.42%	5.16%	12.70%	16.92%	4.21%	17.23%	0.00%
G 9	3.28%	3.47%	21.63%	13.12%	0.42%	3.00%	26.23%	10.83%	2.05%	15.97%	0.00%
	3.43%	9.81%	15.31%	11.06%	0.00%	0.01%	29.28%	12.10%	2.28%	16.72%	0.00%
	2.79%	3.41%	21.27%	14.36%	0.00%	0.01%	26.57%	10.98%	6.97%	13.64%	0.00%
G 10	3.84%	4.50%	20.31%	16.56%	0.68%	0.00%	24.63%	10.07%	3.95%	15.45%	0.00%
	0.57%	12.23%	18.03%	5.75%	0.00%	0.02%	11.52%	31.36%	7.38%	13.15%	0.00%
	3.94%	4.62%	20.83%	14.87%	1.86%	1.11%	24.78%	9.11%	3.70%	15.17%	0.00%



	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}
G 11	2.05%	5.17%	24.01%	19.26%	1.04%	3.85%	10.64%	18.32%	4.51%	11.14%	0.00%
	2.15%	5.41%	25.10%	15.44%	2.03%	0.02%	21.50%	11.98%	4.72%	11.64%	0.00%
	3.61%	3.54%	15.91%	15.59%	2.84%	0.47%	18.67%	12.16%	5.95%	21.26%	0.00%
G 12	3.89%	2.36%	19.02%	10.96%	0.00%	3.06%	31.31%	3.69%	6.72%	18.99%	0.00%
	3.69%	2.23%	20.90%	12.98%	0.00%	0.00%	25.38%	10.47%	6.37%	17.98%	0.00%
	4.88%	1.29%	19.95%	18.23%	0.00%	0.00%	19.57%	15.44%	4.05%	16.60%	0.00%
G 13	3.50%	7.34%	15.44%	12.29%	1.02%	1.06%	26.72%	11.02%	4.47%	17.14%	0.00%
	2.67%	2.16%	22.30%	12.91%	2.60%	2.65%	27.32%	7.72%	3.84%	15.82%	0.00%
	1.54%	11.71%	22.78%	12.73%	2.56%	2.53%	26.12%	7.39%	3.64%	8.99%	0.00%
G 14	2.80%	2.26%	23.37%	11.43%	3.58%	2.18%	23.90%	9.85%	4.03%	16.59%	0.00%
	2.00%	5.14%	23.05%	12.51%	1.03%	1.07%	25.98%	10.71%	3.83%	14.68%	0.00%
	4.89%	5.53%	18.28%	12.66%	1.04%	1.08%	26.28%	10.84%	3.79%	15.61%	0.00%
G 15	4.53%	0.58%	15.64%	16.14%	1.32%	4.89%	29.63%	1.59%	3.48%	22.19%	0.00%
	3.57%	0.29%	20.98%	18.71%	0.00%	2.59%	16.92%	13.28%	6.18%	17.49%	0.00%
	5.03%	3.04%	14.13%	12.86%	2.50%	2.89%	18.93%	14.86%	3.49%	22.27%	0.00%
G 16	2.71%	6.98%	19.69%	11.29%	3.63%	1.09%	24.46%	10.52%	3.82%	15.80%	0.00%
	3.83%	2.82%	22.52%	13.93%	3.30%	2.00%	22.05%	10.04%	4.72%	14.79%	0.00%
	3.85%	2.83%	22.63%	16.72%	6.12%	0.00%	17.49%	11.92%	3.58%	14.86%	0.00%
G 17	4.00%	0.32%	19.25%	15.51%	3.70%	0.02%	28.78%	1.86%	5.51%	21.03%	0.00%
	3.61%	0.29%	21.21%	16.08%	3.83%	2.54%	17.29%	11.21%	6.25%	17.68%	0.00%
	3.66%	0.30%	21.47%	13.61%	2.64%	2.63%	17.22%	14.26%	6.32%	17.89%	0.00%
G 18	4.34%	1.90%	15.86%	15.55%	3.29%	1.99%	22.04%	9.70%	4.00%	21.33%	0.00%
	3.68%	1.61%	20.56%	16.91%	1.33%	2.26%	21.07%	11.12%	3.39%	18.07%	0.00%
	4.93%	2.14%	12.77%	16.71%	2.49%	0.02%	19.07%	12.99%	4.73%	24.15%	0.00%
G 19	6.43%	3.20%	18.38%	16.52%	1.11%	0.00%	28.66%	6.66%	3.11%	15.91%	0.01%
	4.02%	0.32%	23.73%	17.24%	2.95%	2.94%	19.60%	9.88%	3.99%	15.29%	0.03%
	2.49%	1.81%	24.60%	14.87%	2.52%	1.18%	21.63%	9.30%	6.36%	15.23%	0.00%
G 20	3.31%	5.36%	22.78%	12.73%	1.26%	1.16%	26.24%	10.52%	3.36%	13.28%	0.00%
	4.45%	1.94%	16.24%	16.37%	1.24%	0.00%	25.14%	8.70%	4.08%	21.84%	0.00%
	3.50%	0.27%	20.68%	15.02%	1.14%	0.01%	22.75%	12.89%	4.61%	19.12%	0.00%

Table 9 listed the waiting factors for the 11 variables in each generation. Figure 7 through figure 18 demonstrate the change of the weighting factor associated with each variable in each generation.

Table 9 Weight Sets of Maximum Profit from each Generation

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}
G 1	9.16%	2.00%	18.02%	22.87%	2.44%	1.49%	2.50%	15.98%	7.34%	18.15%	0.04%
G 2	2.46%	18.97%	16.83%	18.61%	2.99%	3.24%	2.50%	24.72%	1.90%	7.70%	0.07%
G 3	6.55%	6.40%	14.38%	15.69%	0.12%	0.37%	20.66%	17.08%	1.27%	17.45%	0.03%
G 4	6.55%	3.18%	19.46%	17.93%	1.56%	0.35%	19.97%	14.75%	1.09%	15.15%	0.01%
G 5	4.60%	10.62%	16.63%	14.98%	0.33%	2.43%	24.30%	12.43%	0.91%	12.77%	0.00%
G 6	5.04%	3.47%	21.31%	15.52%	1.54%	0.37%	22.02%	16.24%	1.02%	13.45%	0.01%
G 7	5.28%	3.64%	22.34%	13.86%	6.55%	0.32%	19.74%	13.18%	0.96%	14.11%	0.01%
G 8	2.09%	3.46%	22.11%	10.04%	0.16%	0.47%	3.24%	32.21%	8.28%	17.93%	0.01%
G 9	5.00%	3.44%	21.09%	15.07%	2.61%	0.48%	30.20%	5.94%	1.20%	14.96%	0.00%
G 10	3.03%	4.18%	20.34%	14.98%	9.05%	0.25%	23.54%	0.00%	8.65%	15.96%	0.02%
G 11	3.86%	2.59%	23.77%	21.82%	0.42%	4.30%	17.69%	13.06%	0.82%	11.67%	0.00%
G 12	4.55%	5.07%	19.50%	16.13%	6.23%	0.23%	22.41%	10.03%	0.60%	15.24%	0.00%
G 13	5.13%	2.74%	25.17%	19.88%	4.47%	2.40%	24.47%	5.20%	1.28%	9.23%	0.02%
G 14	0.99%	5.82%	35.79%	17.15%	2.19%	3.14%	20.21%	12.51%	1.61%	0.58%	0.00%
G 15	4.28%	4.79%	19.64%	17.24%	2.36%	0.29%	26.28%	10.04%	0.68%	14.40%	0.00%
G 16	2.50%	4.31%	22.13%	14.31%	3.32%	1.78%	26.99%	4.73%	6.62%	13.30%	0.00%
G 17	2.25%	4.56%	22.62%	10.85%	2.70%	2.02%	22.04%	11.07%	6.51%	15.37%	0.00%
G 18	2.35%	5.36%	33.02%	21.79%	0.32%	2.33%	17.66%	11.12%	5.89%	0.16%	0.00%
G 19	1.81%	3.69%	21.86%	13.99%	5.02%	4.14%	18.14%	9.12%	6.16%	16.06%	0.00%
G 20	3.25%	3.34%	22.40%	9.95%	2.47%	2.23%	24.43%	11.40%	5.37%	15.15%	0.00%

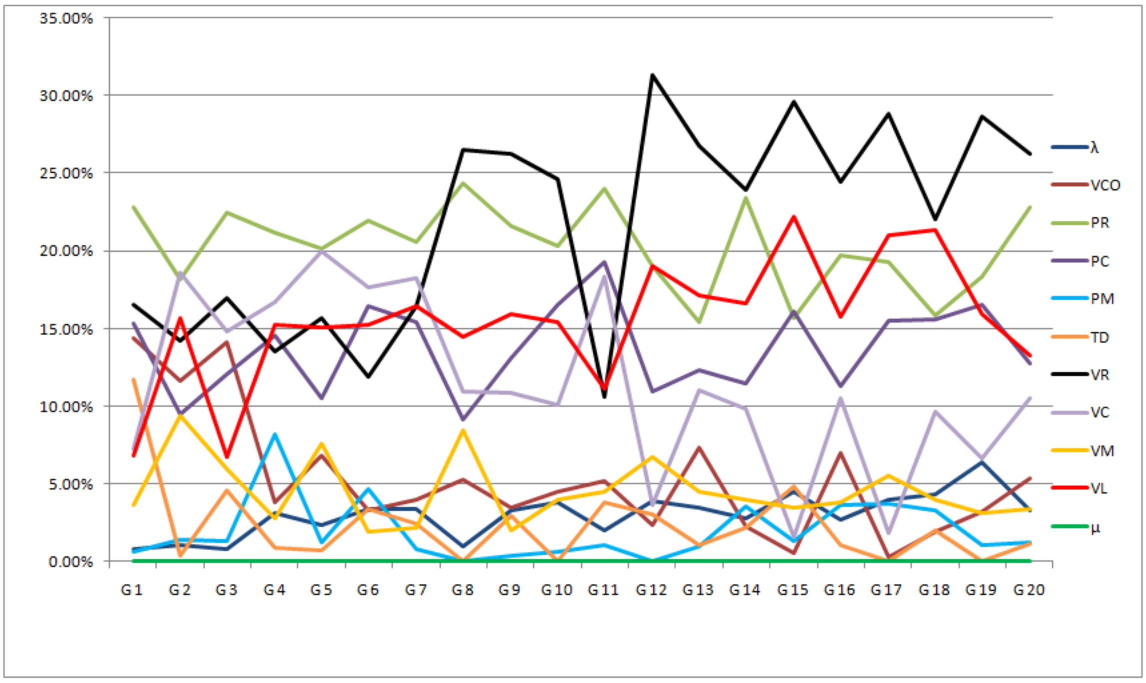


Figure 7: Weight of each Variable Throughout 20 Generations

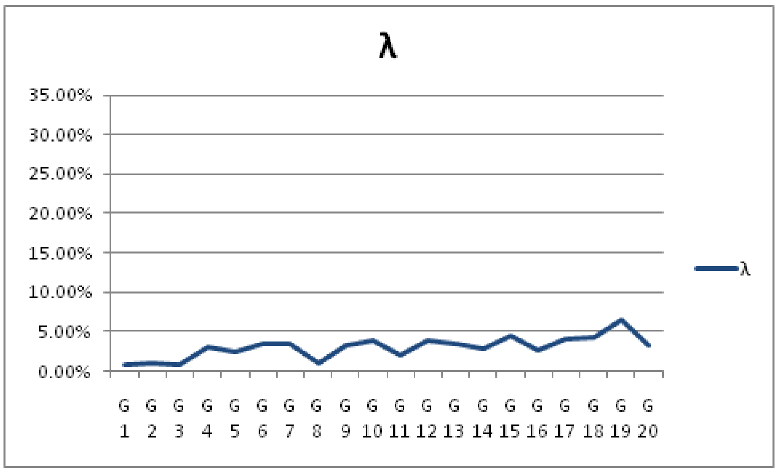


Figure 8: Weight Change of Arrival Rate

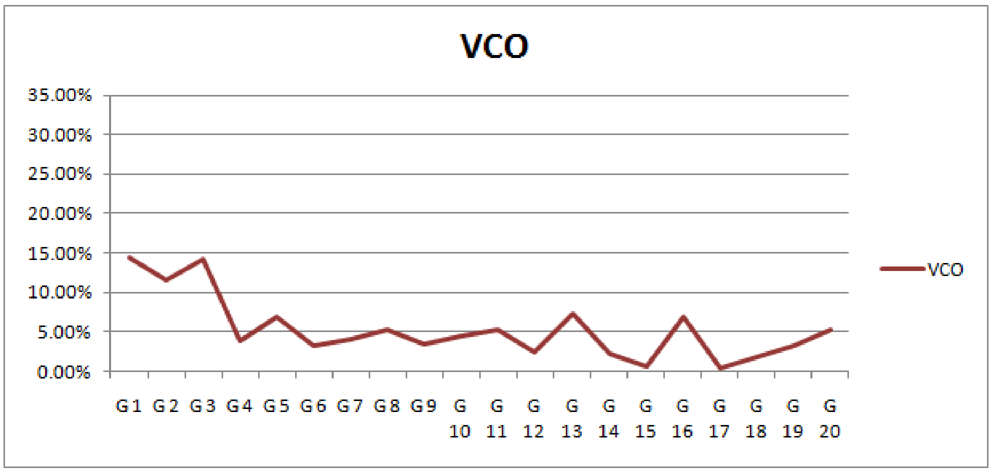


Figure 9: Weight Change of Collection Cost

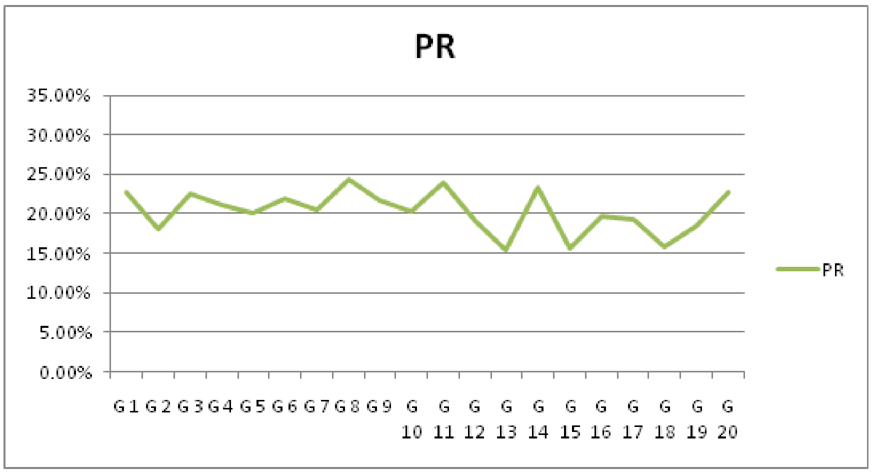


Figure 10: Weight Change of Refurbish Rate

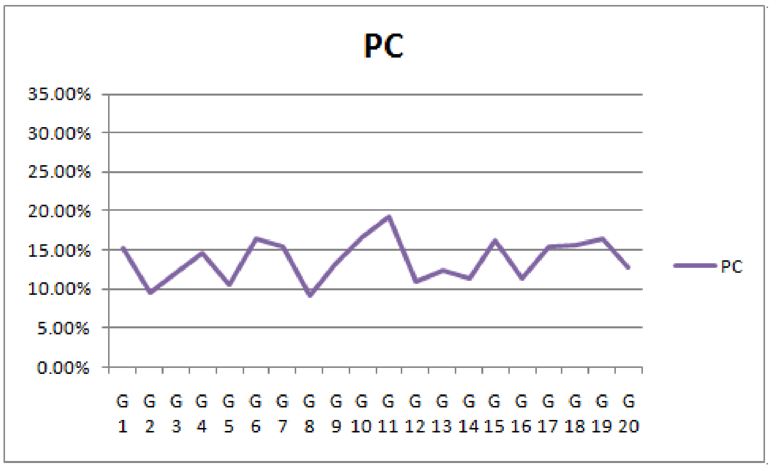


Figure 11: Weight Change of Component Reuse Rate

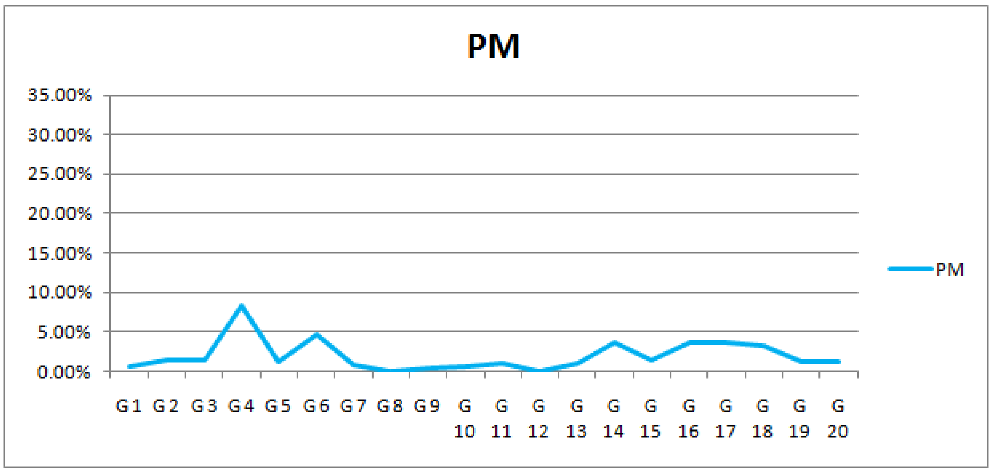


Figure 12: Weight Change of Material Recycle Rate

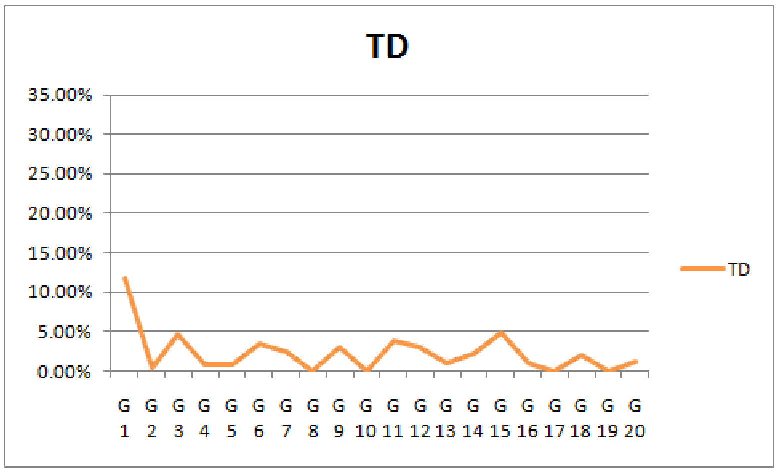


Figure 13: Weight Change of Due Time

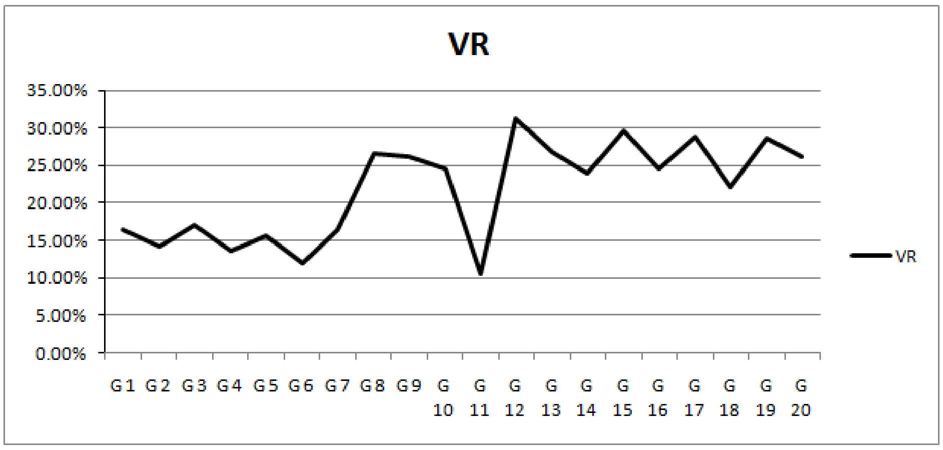


Figure 14: Weight Change of Resale Values

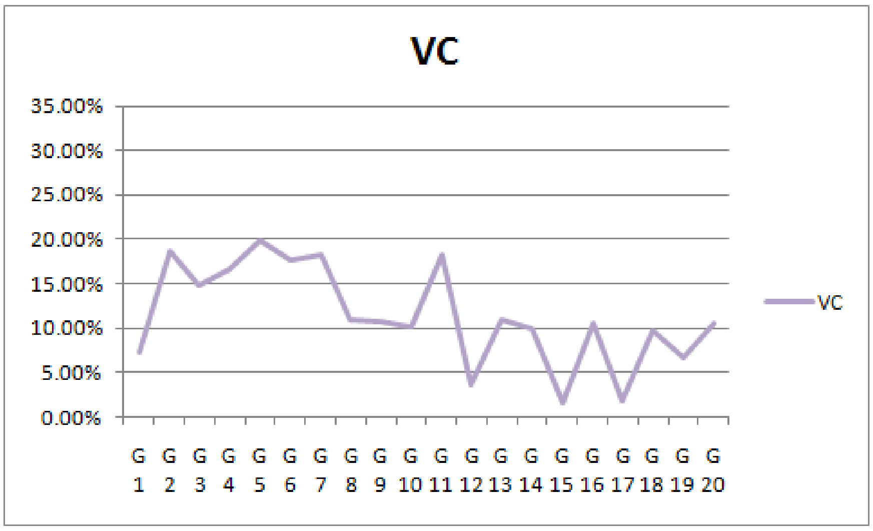


Figure 15: Weight Change of Component Reuse Value

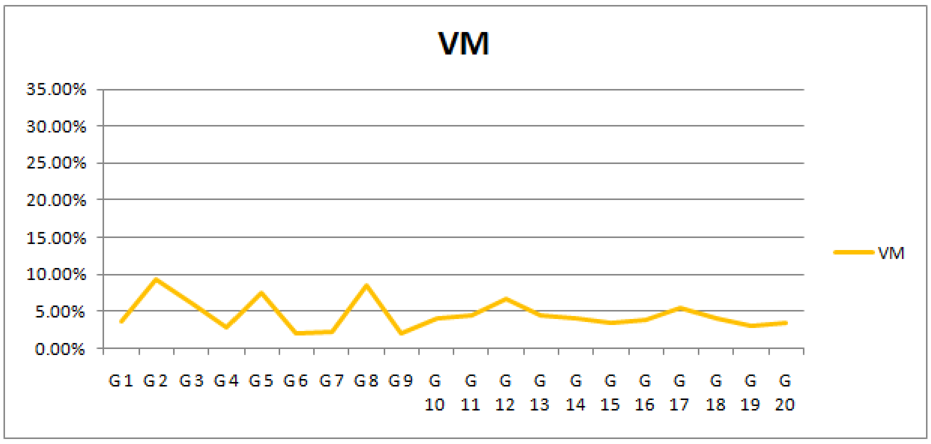


Figure 16: Weight Change of Material Recycle Value

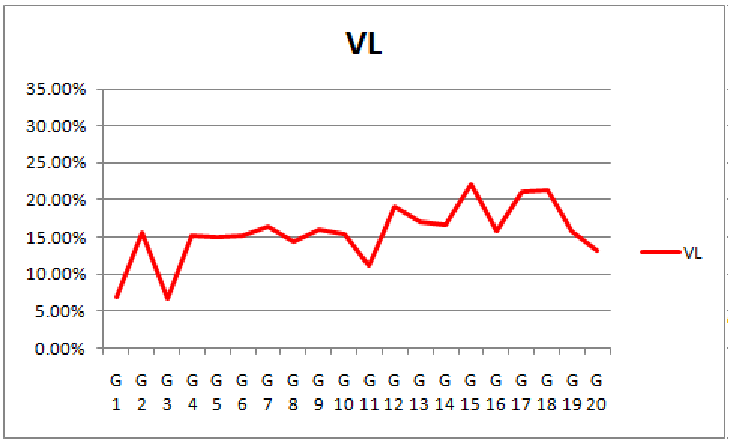


Figure 17: Weight Change of Late Penalty

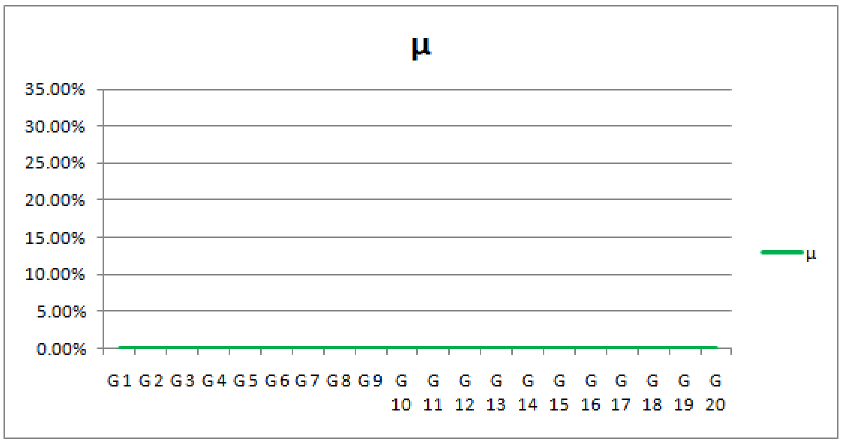
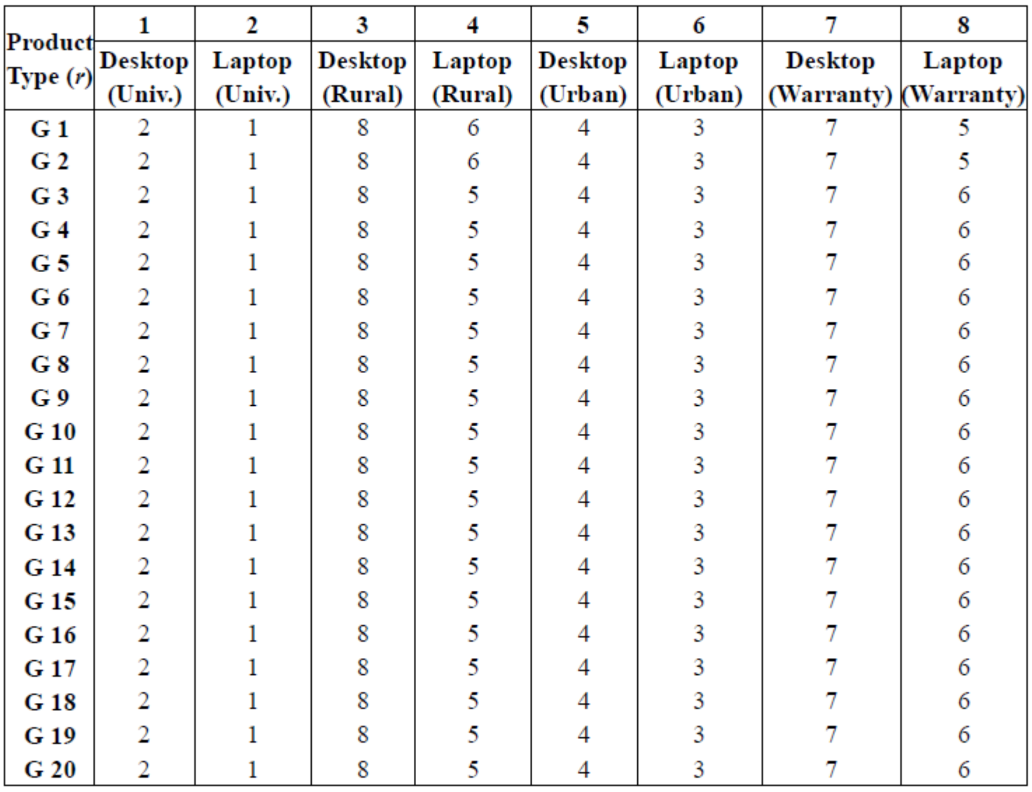


Figure 18: Weight Change of Service Rate

Optimal priority of 8 types of products from each generation is listed as below in Table 10, as shown in Figure 19 optimal priority begins to stay stable from generation 4.

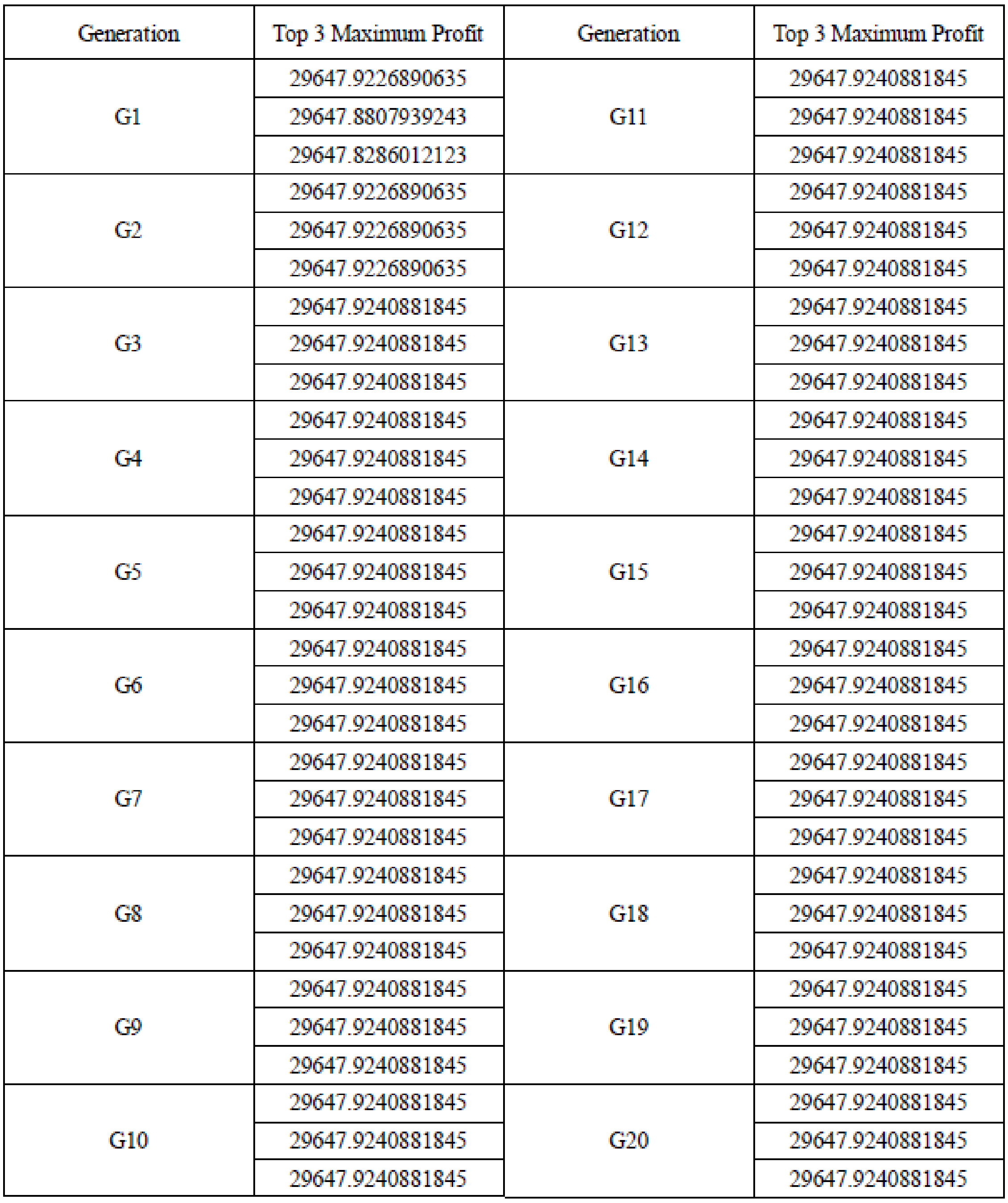
Laptop from university return is of 1^{st} priority, desktop from university is of 2^{nd} priority, desktop from urban area is of 3^{rd} priority, desktop of urban area is of 4^{th} priority, laptop from rural area is of 5^{th} priority, laptop of warranty return is of 6^{th} priority, desktop of warranty return is of 7^{th} priority and desktop from rural area is of 8^{th} priority, and desktop from rural area is of 8^{th} priority.

Table 10 Optimal Priority of Products from each Generation



Product Type (*r*)	1	2	3	4	5	6	7	8
	Desktop (Univ.)	Laptop (Univ.)	Desktop (Rural)	Laptop (Rural)	Desktop (Urban)	Laptop (Urban)	Desktop (Warranty)	Laptop (Warranty)
G 1	2	1	8	6	4	3	7	5
G 2	2	1	8	6	4	3	7	5
G 3	2	1	8	5	4	3	7	6
G 4	2	1	8	5	4	3	7	6
G 5	2	1	8	5	4	3	7	6
G 6	2	1	8	5	4	3	7	6
G 7	2	1	8	5	4	3	7	6
G 8	2	1	8	5	4	3	7	6
G 9	2	1	8	5	4	3	7	6
G 10	2	1	8	5	4	3	7	6
G 11	2	1	8	5	4	3	7	6
G 12	2	1	8	5	4	3	7	6
G 13	2	1	8	5	4	3	7	6
G 14	2	1	8	5	4	3	7	6
G 15	2	1	8	5	4	3	7	6
G 16	2	1	8	5	4	3	7	6
G 17	2	1	8	5	4	3	7	6
G 18	2	1	8	5	4	3	7	6
G 19	2	1	8	5	4	3	7	6
G 20	2	1	8	5	4	3	7	6

Table 11 Top 3 Maximum Profit from each Generation



Generation	Top 3 Maximum Profit	Generation	Top 3 Maximum Profit
G1	29647.9226890635	G11	29647.9240881845
	29647.8807939243		29647.9240881845
	29647.8286012123		29647.9240881845
G2	29647.9226890635	G12	29647.9240881845
	29647.9226890635		29647.9240881845
	29647.9226890635		29647.9240881845
G3	29647.9240881845	G13	29647.9240881845
	29647.9240881845		29647.9240881845
	29647.9240881845		29647.9240881845
G4	29647.9240881845	G14	29647.9240881845
	29647.9240881845		29647.9240881845
	29647.9240881845		29647.9240881845
G5	29647.9240881845	G15	29647.9240881845
	29647.9240881845		29647.9240881845
	29647.9240881845		29647.9240881845
G6	29647.9240881845	G16	29647.9240881845
	29647.9240881845		29647.9240881845
	29647.9240881845		29647.9240881845
G7	29647.9240881845	G17	29647.9240881845
	29647.9240881845		29647.9240881845
	29647.9240881845		29647.9240881845
G8	29647.9240881845	G18	29647.9240881845
	29647.9240881845		29647.9240881845
	29647.9240881845		29647.9240881845
G9	29647.9240881845	G19	29647.9240881845
	29647.9240881845		29647.9240881845
	29647.9240881845		29647.9240881845
G10	29647.9240881845	G20	29647.9240881845
	29647.9240881845		29647.9240881845
	29647.9240881845		29647.9240881845

As shown in Table 11, the maximum profit increase from 29647.8286012123 to 29647.924081845 and stabilizes from the third generation. The top three profits shown above are the maximum found over the 5,000 random generated weight factors which is the reason there does not seem to be a big increase in max profit found over generations. However, this can be explained due to the only change being the switch of two consecutive products.

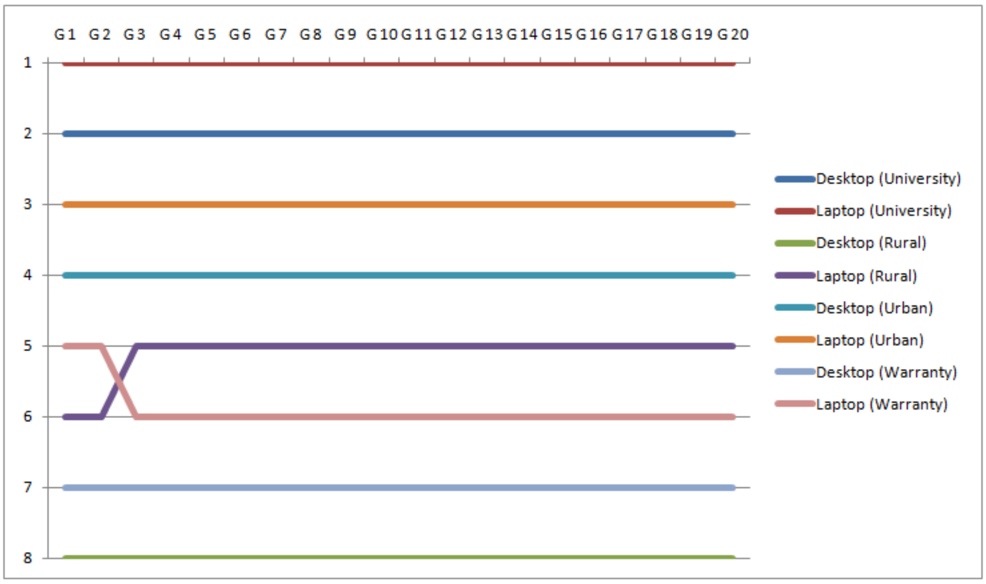


Figure 19: Optimal Products Priority from each Generation

Data obtained from running the Genetic Algorithm on product data shows the following aspects about product information as weights tend to stabilize:

1- Laptops have greater priority than Desktops.

2- University computers have greater priority than urban computers which are greater

than rural computers.

3- Warranty computers have low priority seems to only be affected by due time

In the case of university computers, 90% of the revenue comes from refurbish value and 10% from component value in which both values are greater in university laptops compared to university desktops. In the case of urban and rural computers, calculating the daily revenue shows laptops generate greater profit than desktops with their respective area.

Data used showed revenue from refurbished computers at least three fold from computer component value which is approximately twice as much as material cost. University computers have greater refurbish rate than combined refurbish and component rate from computers found in urban and rural areas. Similarly, computers from urban areas have a greater refurbish rate than combined refurbish and component rate from computers in rural areas.

Computers to be returned because of warranty do not generate profit, however, does cause a loss of money from late penalty. Regardless, data showed these products to be at the end of the priority list, only having late penalty affect the priority greatly.

Analysis of variables by importance after observed convergence reveals following order

Order of importance:

$$VR > PR > VL > PC > VC > VCO > VM > \lambda > PM > TD > \mu$$

Weight ratios between refurbish, component and material value was approximately equal to their respective rates, which seems accurate since increasing the rate by some percentage will give the same revenue as increasing the value respective to it. However, increasing refurbish, component and material rates decreases profit by increasing depreciation costs which is why refurbish, component and material values tend to be greater than their respective rates when increasing generations populated.

The order of of the following variables that will be discussed are respective to their weight importance.

Between refurbish, component and material resale value and rate, it is easily observable that refurbish variables are more influential than component variables which impact more than material variables describing the product. This is because the value for each variable is greater approximately by twice the amount respectively to their increasing influence.

Apart from refurbish, component and material variables, it can be seen that late penalty has a high priority which can be explained by the exponential component being multiplied to it.

Next is λ, which should have a high priority because it is a scalar multiplier to refurbish, component and material variables; however, it is also a scalar multiplier to losses from depreciation which decreases its importance to be big.

The rest of the variables only affect losses by collection costs, late penalties and depreciation starting with collection cost.

Late penalty is second to last in variable importance which can be explained by it being a scalar multiple as well as the lack of late penalty in six out of the eight products in the products observed.

μ affects the equation linearly and only by a factor of less than 1%, which causes it to be the least important when finding the priority index for a given product.

Chapter VI

Conclusion and Future Research

This thesis work focused on optimization of priority based production planning for remanufacturing. Fifteen variables that can affect the performance of a remanufacturing system were identified and investigated in this work. A normalization process was used to covert factors related to these issues so that a weighting factor can be applied to generate a priority index, which was later used to compute the priority assignments. A profit function is derived from based on the waiting time of the products using priority Queueing theory. The impact of fifteen variables impact on cost and revenue are captured in this function. Genetic Algorithm is used to find the optimal waiting factors, which reflect the importance of the issues that need to be considered in determination of the priority dispatching rule..

A case study was conducted to validate the methodology. A remanufacturing facility located in Austin Texas was selected and data were collected. The cost related to holding and depreciation dependent on other variables, hence the number of variables was reduced to eleven. The result from the case study indicated that the method is effective and time-efficient. It can provide support to decision making in production planning in remanufacturing. The result and the optimal weighting factors can also be applied to similar business.

For other types of remanufacturing systems, this method can be easily adapted and applied. The weighting factors can also be used which can provide a close to optimal solution in product priority assignment.

It shall be noted that this research has made assumptions to simplify to the manufacturing system for the ease of analysis. More advanced model that considers other parameters might deserve effort. Hence future research work are suggested as follows:

1)

This research only considered single arrival with priority. In reality, products with different type arrive in bulk. A more advanced Queueing model that considers baulk arrival would be desired.

2)

In a real remanufacturing system, a run size is usually used to process a specific type of return product. Before reaching the run size, the production will continue on the same products, even though higher priority products are waiting in a queue. A suitable Queueing model that incorporate production run size is recommended.

3)

This research work focused on the optimization of the priority, other decision variables, such as workforce level, production line design are assumed to be fixed and given. Future research is suggested so that these decision variables can be modeled all together and optimized at the same time.

4)

Due to lack of data, the case study only considered eight types of product. A remanufacturing facility generally can process much more products. Future work in data collection is recommend so that the algorithm can executed to reveal more insights for remanufacturing planning.

References

V. Damiel R. Guide Jr., Mark E. Kraus, Rajesh Srivastava, Scheduling policies for remanufacturing, Int. J. Production Economics 48 (1997) 187-204

V. Daniel R. Guide, Mark E. Kraus, Rajesh Srivastava, Scheduling policies for remanufacturing, International Journal of Production Economics 48 (2), 1997, pp: 187-204

Priority dispatching with operation due dates in a job shop, John J. Kanet, Jack C. Hayya, Journal of Operations Management, Volume 2, Issue 3, May 1982, Pages 167-175

Simple heuristics for push and pull remanufacturing policies, Erwin A. van der Laan, Ruud H. Teunter, European Journal of Operational Research 175 (2006) 10841102.

Inventory control for a MARKOVIAN remanufacturing system with stochastic decomposition process, Katsuhiko Takahashi, Katsumi Morikawa, Myreshka, Daisuke Takeda, Akihiko Mizuno, International Journal of Production Economics, Volume 108, Issues 1-2, July 2007, Pages 416-425

On optimal inventory control with independent stochastic item returns, Moritz Fleischmann, Roelof Kuik, European Journal of Operational Research, Volume 151, Issue 1, 16 November 2003, Pages 25-37

Periodic review, push inventory policies for remanufacturing, B. Mahadevan, David F. Pyke, Moritz Fleischmann, European Journal of Operational Research, Volume 151, Issue 3, 16 December 2003, Pages 536-551

An investigation of lead-time effects in manufacturing/ remanufacturing systems under simple PUSH and PULL control strategies

Lead time effects and policy improvement for stochastic inventory control with remanufacturing, Karl Inderfurth, Erwin van der Laan, International Journal of Production Economics, Volume 71, Issues 1-3, 6 May 2001, Pages 381-390

Optimal policies in hybrid manufacturing/remanufacturing systems with product substitution, Karl Inderfurth, International Journal of Production Economics, Volume 90, Issue 3, 18 August 2004, Pages 325-343

A new approach for controlling a hybrid stochastic manufacturing/remanufacturing system with inventories and different leadtimes, G. P. Kiesmller

Inventory optimization in a one product recoverable manufacturing system, S. Sebnem Ahiska, Russell E. King, International Journal of Production Economics, In Press, Corrected Proof, Available online 1 November 2009

A hybrid simulation optimization method for production planning of dedicated remanufacturing, Jianzhi Li, Miguel Gonzlez, Yun Zhu, International Journal of Production Economics, Volume 117, Issue 2, February 2009, Pages 286-301

Production planning and control for remanufacturing: industry practice and research needs. V. Daniel R. Guide Jr., Journal of Operations Management, Volume 18, Issue 4, June 2000, Pages 467-483

Production planning and control for remanufacturing: a state-of-the-art survey, V. Daniel R. GuideJr., Vaidyanathan Jayaraman, Rajesh Srivastava, Robotics and Computer-Integrated Manufacturing, Volume 15, Issue 3, June 1999, Pages 221-230

Production planning and inventory control with remanufacturing and disposal, Erwin van der Laan, Marc Salomon, European Journal of Operational Research, Volume 102, Issue 2, 16 October 1997, Pages 264-278

Product recovery in stochastic remanufacturing systems with multiple reuse options, K. Inderfurth, A. G. de Kok, S. D. P. Flapper, European Journal of Operational Research, Volume 133, Issue 1, 16 August 2001, Pages 130-152

Models for production planning under uncertainty: A review, J. Mula, R. Poler, J.P. Garca-Sabater, F.C. Lario, International Journal of Production Economics, Volume 103, Issue 1, September 2006, Pages 271-285

Uncertainty management in optimal disassembly planning through learning-based, Christos Zikopoulos, George Tagaras, European Journal of Operational Research, Volume 182, Issue 1, 1 October 2007, Pages 205-225

A production rate control policy for stochastic repair and remanufacturing systems, Robert Pellerin, Javad Sadr, Ali Gharbi, Roland Malham, International Journal of Production Economics, Volume 121, Issue 1, September 2009, Pages 39-48

Quantitative models for reverse logistics: A review, Moritz Fleischmann, Jacqueline M. Bloemhof-Ruwaard, Rommert Dekker, Erwin van der Laan, Jo A. E. E. van Nunen, Luk N. Van Wassenhove, European Journal of Operational Research, Volume 103, Issue 1, 16 November 1997, Pages 1-17

Controlling inventories with stochastic item returns: A basic model, Moritz Fleischmann, Roelof Kuik, Rommert Dekker, European Journal of Operational Research, Volume 138, Issue 1, 1 April 2002, Pages 63-75

An inventory model with dependent product demands and returns, Gudrun P. Kiesmller, Erwin A. van der Laan, International Journal of Production Economics, Volume 72, Issue 1, 30 June 2001, Pages 73-87

Heuristics for the economic lot scheduling problem with returns, Ruud Teunter, Ou Tang, Konstantinos Kaparis, International Journal of Production Economics, Volume 118, Issue 1, March 2009, Pages 323-330

Gross, D., Harris, C., 1998. Fundamentals of Queueing Theory, John Wiley & Sons, Inc., 3rd Edition.

Donald Gross, John F. Shortle, James M. Thomson, Carl M. Harris, Fundamentals of Queueing Theory 4th Edition.

Walraevens, Joris., Wittevrongel, Sabine., Bruneel, Herwing., A Discrete-Time priority queue with train arrivals. Stochastic Models, 23:489-512, 2007.

Walraevens, Joris., Steyaert, Bart., Bruneel, Herwing., A preemptive repeat priority queue with resampling: Performance analysis. Ann Oper Res (2006) 146: 189-202.

Choi, Doo Il., Kim, Tae-Sung., Le, Sangmin., Analysis of a queueing system with a general service scheduling, with applications to telecommunications network traffic control. European Journal of Operational Research 178 (2007) 463-471.

Chelst, K., Tilles, A.Z., and Pipis, J.S., 1981. A Coal Unloader: A Finite Queueing System With Breakdowns, The Institute of Management Science (INTERFACES), vol. 11, No. 5, October 1981.

Appendix A

Appendix A

Genetic Algorithm Code

File Name: Main.java

Main.java 8/6/10 2:16 PM

```
import java.io.IOException;

public class Main{
    public static void main(String [] args)throws IOException{
        // Starts the program
        // This is done to bypass the static class Main
        Genetic main = new Genetic();
    }
}
```

File Name: Genetic.java

```
import java.util.Scanner;
import java.io.FileReader;
import java.io.IOException;

public class Genetic
{

    Inventory inventory;

    public Genetic()throws IOException
    {/*
        // Number of Weights, Population Size
        Population test = new Population(13,10000);

        // Range of Summation of Weights (Min,Max)
        test.setCRange(0,91);

        // Range of a Randomized Weight  (Min,Max)
        test.setRRange(1,150);

        //  1- Weights passed on next generation
        //  2- Fixed Probability of Step Reproduction
        //  3- Step Size
        test.crossStats(0.001,0.50,0.02);

        test.spawn();*/

        readFile("data.txt");

        sample(1,20);
    }

    public void sample(int populations,int generations){
        double min = 0;
        boolean minCheck = false;
        double avg = 0.0;

        for(int i=0;i<populations;i++){
            Population test = new Population(11,10000,inventory);
            test.setCRange(0,1);
            test.setRRange(1,1000);
            test.crossStats(0,0.5,0.02);
            test.spawn();

            for(int j=0;j<generations;j++){
                test.reproduce();
                test.output();
            }
            System.out.println("\n--------------------------------\n");

        }
    }
```

```
        public void readFile(String file) throws IOException {
            int size = 0;
            FileReader in = new FileReader(file);
            Scanner scan = new Scanner(in);

            while(scan.hasNextLine()){
                scan.nextLine();
                size++;
            }

            in.close();

            FileReader in2 = new FileReader(file);

            inventory = new Inventory(size);
            size = 0;

            Scanner scan2  = new Scanner(in2);
            double[] input = new double[11];

            while(scan2.hasNext()){
                if (scan2.hasNextDouble()) {
                    input[size] = scan2.nextDouble();
                    size++;
                    if(size==11){
                        inventory.insert(input);
                        size=0;
                    }
                }
            }

            in2.close();
        }
    }
```

File Name: Population.java

```
import java.util.Random;
import java.lang.Math;
import java.util.SortedMap;
import java.util.TreeMap;
import java.util.Iterator;
import java.text.DecimalFormat;

public class Population{

    Inventory inventory;

    public int cSize;
    public int pSize;
    public int cMin;
    public int cMax;
    public int rMin;
    public int rMax;

    public int elite;
    public int fixedC;
    public int step;

    public int[][] pool;
    public int[][] best;

    public double[] wt;

    public boolean[] checkFit;
    public double[] fit;
    public SortedMap<Double,Integer> sortedFit;
    public double[] bestFit;
    public int[][] bestSort;
    public double totalFit;

    public Population(int x,int y,Inventory inv){
        inventory = inv;

        // cSize - Chromosome size
        // pSize - Population size
        cSize = x;
        pSize = y;
        elite = 0;

        pool = new int[y][x];
        best = new int[3][x];
        bestFit = new double[3];
        bestSort = new int[3][8];

        wt = new double[x];

        checkFit = new boolean[3];
        checkFit[0] = checkFit[1] = checkFit[2] = false;
        bestFit[0] = bestFit[1] = bestFit[2] = 0;
        fit = new double[y];
        sortedFit = new TreeMap<Double,Integer>();
        totalFit = 0;
    }

    public void setCRange(int a,int b){
        cMin = a;
        cMax = b;
    }

    public void setRRange(int a,int b){
        rMin = a;
        rMax = b;
    }

    public void crossStats(double a,double b,double c){
        elite  = (int) (a*pSize);
        fixedC = (int) (b*pSize);
        step   = (int) (c*pSize);

        if(step == 0)
            step = 1;

        if(step*(step-1) > pSize)
            step = (int) Math.sqrt(pSize);

    }

    public void spawn(){
        Random gen = new Random();
        int tmp;

        for(int i=0;i<pSize;i++){

            for(int j=0;j<cSize;j++){
                tmp = gen.nextInt(rMax)+1;
                pool[i][j] = tmp;
            }

            totalFit += fitness(i);
        }
    }

    public double fitness(int x){
        double ret = 0;
        double totalWt= 0;

        for(int i=0;i<cSize;i++)
            totalWt += pool[x][i];

        for(int i=0;i<cSize;i++){
            wt[i] = (pool[x][i]*cMax)/totalWt;
        }
        ret = f();

        for(int i=0;i<3;i++){
            if(bestFit[i] > ret || !checkFit[i]){
                checkFit[i] = true;
                best[i] = pool[x];
                bestFit[i] = ret;
                bestSort[i] = inventory.prior;
                i = 4;
            }
        }

        for(int i=0;i<cSize;i++){
            pool[x][i] = (int) ((rMax*10*pool[x][i])/totalWt);
        }

        fit[x] = ret;
        sortedFit.put(ret,x);
        return ret;
    }

    public double f(){
        inventory.setWeights(wt);
        return -inventory.totalProfit();

    }

    public void reproduce(){
        int[][] pool2 = new int[pSize][cSize];
        Iterator it = sortedFit.keySet().iterator();
        checkFit[0] = checkFit[1] = checkFit[2] = false;
        int pos = 0;
        int tmp;

        for(int i=0;i<elite;i++){
            if(it.hasNext()){
                tmp = sortedFit.get(it.next());
                pool2[pos] = pool[tmp];
                pos++;
            }
        }

        Random gen = new Random();
        Iterator it2 = sortedFit.keySet().iterator();
        int[] hold = new int[step];

        for(int i=0;i<step;i++){
            if(it2.hasNext()){
                tmp = sortedFit.get(it2.next());

                for(int j=0;j<i;j++){
                    hold[i-j] = hold[i-j-1];
                }

                hold[0] = tmp;

                for(int j=1;j<=i;j++){
                    pool2[pos] = cross(hold[j],tmp,gen.nextInt(cSize/2));
                    pos++;
                    pool2[pos] = cross(tmp,hold[j],gen.nextInt(cSize/2));
                    pos++;
                }
            }
        }

        for(int i=(step*(step-1));i<(fixedC/(2*step));i++){
            if(it2.hasNext()){
                tmp = sortedFit.get(it2.next());

                for(int j=0;j<(step-1);j++){
                    hold[step-j-1] = hold[step-j-2];
                }

                hold[0] = tmp;

                for(int j=1;j<step;j++){
                    pool2[pos] = cross(hold[j],tmp,gen.nextInt(cSize/2));
                    pos++;
                    pool2[pos] = cross(tmp,hold[j],gen.nextInt(cSize/2));
                    pos++;
                }
            }
        }

        for(int i=pos;i<pSize-1;i+=2){
            int a = roulette(gen.nextInt(100000));
            int b = roulette(gen.nextInt(100000));
            pool2[pos] = cross(a,b,gen.nextInt(cSize/2));
            pos++;
            pool2[pos] = cross(b,a,gen.nextInt(cSize/2));
            pos++;
        }

        pool = pool2;
        totalFit = 0;
        sortedFit.clear();

        for(int i=0;i<pSize;i++){
            totalFit += fitness(i);
        }
    }

    public int roulette(int x){
        double stop = x/100000.0;
        double max = sortedFit.lastKey()+1;
        stop = (max*pSize-totalFit)*stop;

        double spin = 0.0;
        int ret = -1;

        while(spin<stop){
            ret++;
            spin += (max-fit[ret]);
        }

        if(ret == -1)
            return 0;

        return ret;
    }

    public int[] cross(int a,int b,int pivot){
        int[] ret = new int[cSize];

        for(int i=0;i<pivot;i++)
            ret[i] = pool[a][i];

        for(int i=pivot;i<(cSize/2+pivot);i++)
            ret[i] = pool[b][i];

        for(int i=(cSize/2+pivot);i<cSize;i++)
            ret[i] = pool[a][i];

        ret = mutate(ret);

        return ret;
    }

    public int[] mutate(int[] tmp){
        Random gen = new Random();
        int chance = gen.nextInt(cSize*1000);

        if(chance < cSize*10){
            if(chance < cSize*5){
                if(chance < cSize){
                    tmp[chance%cSize] = tmp[chance%cSize]/(gen.nextInt(rMax
                        /10)+1);
                    tmp[(chance+1)%cSize] = tmp[(chance+1)%cSize]/(gen.
                        nextInt(rMax/10)+1);
                    tmp[(chance+2)%cSize] = tmp[(chance+2)%cSize]/(gen.
                        nextInt(rMax/10)+1);
                }
                else{
                    tmp[chance%cSize] = tmp[chance%cSize]/(gen.nextInt(rMax
                        /10)+1);
                    tmp[(chance+1)%cSize] = tmp[(chance+1)%cSize]/(gen.
                        nextInt(rMax/10)+1);
                }
            }
            else{
                tmp[chance%cSize] = tmp[chance%cSize]/(gen.nextInt(rMax/10)
                    +1);
            }
        }

        return tmp;
    }

    public void output(){
        String tmp = "";
        int sumW = 0;
        double[] out = new double[cSize];

        for(int j=0;j<3;j++){
            sumW = 0;

            for(int i=0;i<cSize;i++)
                sumW += best[j][i];

            for(int i=0;i<cSize;i++)
                out[i] = round((100.0*best[j][i])/sumW);

            tmp += -bestFit[j];
            tmp += " - \t";
            for(int i=0;i<cSize-1;i++){
                if(out[i]==0)
                    tmp+="0";
                else
                    tmp += out[i];

                tmp += "\t";
            }
            tmp += out[cSize-1];
            tmp += "\n";
        }
        tmp+="\n";
        for(int j=0;j<3;j++){
            for(int i=0;i<8;i++){
                tmp += bestSort[j][i];
                tmp += " ";
            }
            tmp+="\n";
        }
        System.out.println(tmp);
    }

    /*
    public void output(int x){
        String tmp = "";

        tmp += fit[x];
```

```
            tmp += " - (";
            for(int i=0;i<cSize-1;i++){
                tmp += pool[x][i];
                tmp += ",";
            }
            tmp += best[cSize-1];
            tmp += ")";

            System.out.println(tmp);
        }
         */

        double round(double d) {
            DecimalFormat tmp = new DecimalFormat("#.##");
            return Double.valueOf(tmp.format(d));
        }

    }
```

File Name: Inventory.java

```
import java.util.SortedMap;
import java.util.TreeMap;
import java.util.Iterator;
import java.lang.Math;

public class Inventory
{
    int size;
    int current;
    Product[] data;
    double[] weight;
    int[] prior;
    double[] min;
    double[] max;
    boolean[] minCheck;
    boolean[] maxCheck;

    double sum1;
    double sigma;

    public SortedMap<Double,Integer> priority;

    public Inventory(int s){
        size = s;
        current = 0;
        min = new double[11];
        max = new double[11];
        minCheck = new boolean[11];
        maxCheck = new boolean[11];
        data = new Product[s];
        weight = new double[15];

        sum1 = 0;
        sigma= 0;

        for(int i=0;i<11;i++)
            minCheck[i] = maxCheck[i] = false;

        priority = new TreeMap<Double,Integer>();
    }

    public void setWeights(double[] w){
        for(int i=0;i<11;i++)
            weight[i] = w[i];
    }

    public void insert(double[] input){
        int worker = 106;

        if(current < size){

            data[current] = new Product();
            for(int i=0;i<10;i++){
                data[current].var[i] = input[i];

                if(input[i] < min[i] || !minCheck[i]){
                    minCheck[i] = true;
                    min[i] = input[i];
                }
                else if(input[i] > max[i] || !maxCheck[i]){
                    maxCheck[i] = true;
                    max[i] = input[i];
                }

            }

            if(input[10] < min[10] || !minCheck[10]){
                minCheck[10] = true;
                min[10] = input[10];
            }
            else if(input[10] > max[10] || !maxCheck[10]){
                maxCheck[10] = true;
                max[10] = input[10];
            }

            data[current].var[10] = 0.2*(data[current].var[2]*data[current]
                .var[6]+
                                         data[current].var[3]*data[current]
                                             .var[7]+
                                         data[current].var[4]*data[current]
                                             .var[8])/260.0;
            data[current].var[11] = 0.5*(data[current].var[6])/260.0;
            data[current].var[12] = 0.5*(data[current].var[7])/260.0;
            data[current].var[13] = 0.5*(data[current].var[8])/260.0;
            data[current].var[14] = input[10]*worker;

            current++;
        }
    }

    /*
     data[K].var[A] is the variable B for product K

     A  B

     0  lambda
     1  VCO
     2  PR
     3  PC
     4  PM
     5  TD
     6  VR
     7  VC
     8  VM
     9  VL
     10 VH
     11 VDP
     12 VDC
     13 VDM
     14 mu

     */

    public double totalProfit(){
        priority = new TreeMap<Double,Integer>();
        sum1 = 0;
        sigma= 0;

        Priority();

        Iterator it = priority.keySet().iterator();
        double[] tmp;
        double ret = 0;
        double W;
        int pos;
        prior = new int[priority.size()];

        for(int i=0;i<size;i++){
            if(it.hasNext()){
                pos = priority.get(it.next());
                prior[i] = pos;
                tmp = data[pos].var;

                sum1 += (tmp[0]/(tmp[14]*tmp[14]));

                W = sum1/((1-sigma)*(1-sigma-(tmp[0]/tmp[14])));

                ret += tmp[0]*(tmp[2]*tmp[6]+tmp[3]*tmp[7]+tmp[4]*tmp[8]);

                ret -= tmp[0]*tmp[1];

                ret -= tmp[0]*(W+1.0/tmp[14])*(tmp[2]*tmp[11]+tmp[3]*tmp[12
                    ]+tmp[4]*tmp[13]);

                ret -= tmp[10]*W;

                if(tmp[5]!=0)
                    ret -= tmp[9]*(W+1.0/tmp[14])*Math.pow(Math.E,-(0.0+tmp
                        [5])/(W+1.0/tmp[14]));

                sigma += (tmp[0]/tmp[14]);
            }
        }

        return ret;
    }

    public void Priority(){
            double tmp;
            priority.clear();

            for(int i=0;i<size;i++){
                tmp = 0.0;
                /*
                for(int j=0;j<10;j++){
                    tmp+=data[i].var[j]*weight[j];
                }
                tmp+=data[i].var[14]*weight[10];
                */


                tmp+=(max[0]-data[i].var[0])*weight[0]/(max[0]-min[0]);
                tmp+=(1.0-(max[1]-data[i].var[1])/(max[1]-min[1]))*weight[1];

                for(int j=2;j<5;j++)
                    tmp+=(max[j]-data[i].var[j])*weight[j]/(max[j]-min[j]);

                tmp+=(1.0-(max[5]-data[i].var[5])/(max[5]-min[5]))*weight[5];

                for(int j=6;j<10;j++)
                    tmp+=(max[j]-data[i].var[j])*weight[j]/(max[j]-min[j]);

                tmp+=(max[10]-data[i].var[14])*weight[10]/(max[10]-min[10]);

                /*
                tmp+=(max[0]-data[i].var[0])*weight[0]/(max[0]);
                tmp+=(1.0-(data[i].var[1])/(max[1]))*weight[1];

                for(int j=2;j<5;j++)
                    tmp+=(data[i].var[j])*weight[j]/(max[j]);

                tmp+=(1.0-(data[i].var[5])/(max[5]))*weight[5];

                for(int j=6;j<10;j++)
                    tmp+=(data[i].var[j])*weight[j]/(max[j]);

                tmp+=(data[i].var[14])*weight[10]/(max[10]);
                */
                if(priority.containsKey(tmp))
                    tmp+=0.0000000001;

                priority.put(tmp,i);
            }
        }


    }
```

File Name: Product.java

```
public class Product
{
    double[] var;

    public Product(){
        var = new double[15];
    }
    public Product(double[] var2){

        var = new double[15];

        for(int i=0;i<15;i++)
            var[i] = var2[i];

    }

}
```

Biographical Sketch

Xiong Xiong, was born in Hangzhou, Zhejiang Province, P. R. China on August 21st 1985. She obtained her Bachelors degree of Management in Industrial Engineering from Nanjing University of Aeronautics and Astronautics, Jiangsu Province, P.R. China on the year of 2007. Her enthusiasm for higher education and culture diversity brought her to the United States. Following her undergraduate studies, she also obtained her Masters degree in Manufacturing Engineering from the University of Texas Pan-American.