NEW STRATEGIC AND DYNAMIC VARIATION REDUCTION TECHNIQUES FOR ASSEMBLY LINES

Rami Musa

Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
In
Industrial and Systems Engineering

Dr. Ebru K. Bish
Dr. F. Frank Chen (Chair)
Dr. Michael Deisenroth
Dr. Robert H. Sturges

March, 29th 2007
Blacksburg, Virginia

Keywords: Variation Reduction, Rolled Yield Throughput, Selective Assembly, Inspection Planning, CAD Data, Six Sigma

NEW STRATEGIC AND DYNAMIC VARIATION REDUCTION TECHNIQUES FOR ASSEMBLY LINES

RAMI MUSA

ABSTRACT

Variation is inevitable in any process, so it has to be dealt with effectively and economically. It has been the focal point of many researchers to control, predict or reduce the variation of a product to increase its competitiveness in the current global marketplace.

Reducing variation can be achieved in assembly lines strategically and dynamically. It can be reduced strategically by developing optimal process, setup and inspection plans. It can be reduced dynamically by utilizing real-time data (inspection data) to effectively assign parts together, and to adjust process settings and design requirements accordingly. Implementing both the strategic and dynamic variation reduction techniques is expected to lead to further reduction in the number of failed final assemblies.

The dissertation is divided into three major parts. In the first part, we propose to reduce variation for assemblies by developing efficient inspection plans based on (1) historical data for existing products, or simulated data for newly developed products; (2) Monte Carlo simulation; and (3) optimization search techniques. The cost function to be minimized is the total of inspection, rework, scrap and failure costs. The novelty of the proposed approach is three-fold. First, the use of CAD data to develop inspection plans for newly launched products is new, and has not been introduced in the literature before. Second, frequency of inspection is considered as the main decision variable, instead of considering whether or not to inspect a quality characteristic of a subassembly. Third, we use a realistic reaction plan (rework-scrap-keep) that mimics reality in the sense that not all out-of-tolerance items should be scrapped or reworked. Instead, our reaction plan in the case of out-of-tolerance items is to rework an item if it does not drift significantly

from the required specifications, and to scrap it if it drifts too much from the specifications. The proposed optimization search technique in this work is Genetic Algorithm (GA).

At a certain stage, real-time inspection data for a batch of subassemblies could be available. In the second part of this dissertation, we propose utilizing this data in near real-time to dynamically reduce variation by assigning the inspected subassembly parts together. We call this problem: Dynamic Variation Reduction (DVR). We also propose mating the inspected subassembly items so the rolled-yield-throughput is maximized. This is called the Dynamic Rolled Yield Throughput Maximization (DTM). In proposing mathematical models for these problems, we found that they are hard to solve using traditional optimization techniques. Therefore, we propose using heuristics. We found simple rules that can solve DVR near-optimally and optimally for simple cases. On the other hand, the proposed heuristics for DTM are: (1) simple greedy heuristics, (2) two Simulated Annealing algorithms, and (3) two Ant Colony Optimization algorithms. An approach based on mathematical modeling and heuristic solution is proposed to solve the selective assembly problem more generally than it has been presented previously in the literature.

Finally, we propose exploring opportunities to reduce the aforementioned cost function by integrating the inspection planning model with the Dynamic Throughput Maximization (DTM) model. This hybrid model adds one decision variable in the inspection planning; which is whether to implement DTM (assemble the inspected subassemblies selectively) or to assemble the inspected items arbitrarily. We expect this hybrid implementation to substantially reduce the failure cost when assembling the final assemblies for some cases. To demonstrate this, we solve a numerical example that supports our findings.

Keywords: Variation Reduction, Rolled Yield Throughput, Selective Assembly, Inspection Planning, CAD Data, Six Sigma, Metaheuristic

DEDICATION

To my mother Da’ed, father Adnan and two brothers: Naji and Husam.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my academic advisor, Dr. F. Frank Chen, who believed in me and provided valuable advice to me during my stay at Virginia Tech. Without his support and encouragement, I would certainly not have gone this far in my education. Also, I cannot thank enough Dr. Robert Sturges (Dr. Bob) for his insightful technical contribution to my research. I would also like to thank Dr. Michael Deisenroth and Dr. Ebru Bish for serving on my Ph.D. committee, and for providing valuable input through my proposal, progress meeting, and final defense. Thanks to *Advancing Computing as a Science and Profession* (ACM) for giving me the permission to adapt a copyrighted material (Figures 1.5 and 1.6 in this dissertation). I also would like to thank my friends (in no particular order): Bacel Maddah, Walid Nasr, Shaadi Elswaifi, Waseem Faidi, Mohammad Hamdan, Mohammad Younis, Gokhan Ozgen, Natalie Cherbaka and Leonardo Rivera. I would like to extend a special *thank you* to Ahmed Ghoniem for his sincere help in my work. Finally, I would like to express my deepest appreciation and gratitude to my family for their support and help.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION AND MOTIVATION

Because uncontrolled conditions are inevitable, variation is a part of any manufacturing process, and must be dealt with effectively. Variation may be controlled to be within specified limits, or reduced to a certain acceptable level so a product can be usable. In this chapter; we introduce the background, motivation for, and originality of the work in this dissertation, the concepts and terms used, and the architecture of the dissertation.

Variation can be categorized according to its behavior and source. Its behavior is determined by the probability of achieving a particular value. This can be reasonably represented by using probability distribution functions (PDF) and their parameters. The variation sources could be within a process and/or between processes. A single machine can never produce two similar quality characteristics for a feature. This is what we refer to as within-process variation. Between-process variation is the variation due to variation buildup (stackup) when a product is manufactured through a multi-stage production line.

Manufacturers are striving to reduce variation by implementing new techniques and approaches. Variation reduction in assembly can be achieved by various approaches. Some of these are: (1) using processes with high process capabilities to make subassemblies, (2) designing efficient process and setup plans that reduce the tolerance stackup in subassemblies and final assemblies, (3) implementing efficient inspection plans and corrective actions (keep/rework/scrap) based upon the inspected variation, and, (4) mating subassemblies together in order to achieve the least variation and maximize the rolled-yield-throughput as much as possible. We discuss implementing the third and the forth approaches independently (Chapters 3-6) and then hybridly (Chapter 7) in this dissertation.

1.1. DEFINITIONS AND CONCEPTS

1.1.1 Six Sigma

Six Sigma is a business improvement approach that standardizes steps for solving problems to reduce errors (variations). Generally, there are two Six Sigma methodologies; one is for existing products/ processes, and another is for new products or processes. *DMAIC* (Define, Measure, Analyze, Improve and Control) is a methodology for existing products/processes. *DMADV* (Define, Measure, Analyze, Design and Verify) is a methodology for new products. These steps for solving problems may sound obvious and common-sense; however the methodologies provide standard language to allow people with different educational and cultural backgrounds to interact efficiently. For instance, a Medical Doctor (MD) and a Product Engineer now have a common language to design and produce a medical device. Implementing Six Sigma has proven to result in significant savings for major industrial organizations such as General Electric, Motorola and Honeywell. Thanks to Six Sigma implementation, General Electric's 1998 annual report states: "GE achieved cost savings of more than three quarters of a billion dollars in 1998" [GE Annual Report (1998)]. Problems solved in Six Sigma are *process*- or *product*- related. Product-related problems could refer to a quality characteristic improvement, such as the diameter of a hole. Process related problems refer to a process parameter improvement, such as cycle time for milling a free-form surface. Six Sigma aims to achieve less than 3.4 defects per million.

Now, let us explain what is meant by DMAIC (*Define*, *Measure*, *Analyze*, *Improve* and *Control*) steps:

Define refers to defining customer needs for a product or process to be improved (optimized). For example, a product feature to be studied for improvement is a hole-diameter in a plate. The customer's specification for the diameter is 10±0.10 mm. The existing hole-diameter is found to be 10.2±0.37 mm. The target is to achieve a diameter that outperforms customer specifications by reaching 10±0.05 mm.

Measure refers to determining which characteristics are to be measured, designing the datasets, and then collecting the data. (In our current example, the input-output dataset would be

determined in a brainstorming session by a team of manufacturing engineers and technicians.) The output (Y) from this process is hole-diameter. The inputs (X's) are coolant temperature (X_1), cutting tool life (X_2), and the operator's name (X_3). Full or fractional factorial designs can be used to construct rigorous experiments. In this example, a full factorial design with two levels (2^3 experiments) needs 8 runs (treatment combinations) of experiments. More replications at each treatment combination are usually advised. The levels of the factors have to be numerically defined in this step. For instance, the *low* value for the cutting life cycle could be 0 hours, and the *high* value is 12 hours.

Analyze refers to establishing key inputs and outputs from the relationship to be studied in the previous step using techniques such as regression analysis, Analysis of Variance (ANOVA), One Factor at a Time (OFAT), etc. In this example, after analyzing the data, we have found at this stage that cutting tool life (X_2) is most significant to the quality of the hole-diameter.

Improve refers to identifying improvements in the inputs (X's) to achieve the target by using tools such as Response Surface Methodology (RSM), or other optimization search techniques. In this example, we know by now by using RSM that we can achieve the targeted hole-diameter by replacing the cutting tool every 3.5 hours.

Control refers to monitoring, documenting and assigning tasks to achieve the performance using different tools such as control chart, etc. We can monitor the diameter using control charts to make sure that our diameter is stable and it is within the designed specifications.

1.1.2 Rolled Yield Throughput (Y_{RT})

Process capability, *process capability with drift* and *rolled yield throughput* are metrics that are widely used in Six Sigma literature. We focus on using the rolled yield throughput in this work. Rolled yield throughput (Y_{RT}) can be defined as the probability that a product will satisfy all the prescribed tolerances. It can be mathematically represented as follows:

$$Y_{RT} = \int_{L_1}^{U_1}\int_{L_1}^{U_2}\ldots\int_{L_m}^{U_m} P(q_1, q_2, \ldots, q_m)dq_1 dq_1 \ldots dq_m \qquad (1.1)$$

Where $P(q_1, q_2, \ldots, q_m)$ is the joint probability density function (PDF) for the quality characteristics; q's and L_i and U_i refer to the lower and upper specification limits for a quality characteristic i. This is often to be maximized throughout this work (Chapters 5-7). If the m quality characteristics were found to be independent, then Equation 1.1 will reduce to Equation 1.2.

$$Y_{RT} = \prod_{i=1}^{m}\int_{L_i}^{U_i} P(q_i)dq_i \qquad (1.2)$$

1.1.3 Genetic Algorithm (GA)

Genetic Algorithm is a metaheuristic that mimics Darwin's evolutionary theory. It has been proven to be an effective in solving difficult combinatorial optimization problems. Some important terms in GA are: chromosome, gene, population, crossover, mutation, fitness, parent, offspring, and elitism. Solutions in GA are generally represented by a population of chromosomes (strings) of discrete numbers (genes), e.g. 0110001. Our goal is to optimize (maximize or minimize) an objective function (fitness). We can summarize the process for an unconstrained optimization problem with an example as follows:

(1) Generate initial population of solutions (chromosomes): we usually begin with a population of randomly generated solutions. The size of population (P) impacts the convergence toward the optimal solution. We refer to the number of genes in each chromosome by N.

(2) Fitness evaluation: evaluate the fitness for each chromosome in the population, rank and sort them accordingly.

(3) Selection process: we select $P/2$ pair of chromosomes (parents) in order to generate the offsprings. Each pair generates two offsprings. The fittest chromosomes (the ones with the

highest objective function values in the case of maximization) will have more chance (probability) to be selected to be paired. This can be achieved using the *rank selection approach* by associating each chromosome with the following probability: $P_n = (P - n + 1) / \sum_{i=1}^{P} i$, where n is the chromosome rank. For example; if we have 6 chromosomes (P=6), then the probabilities of mating for the ranked chromosomes according to their fitness values are: 28.6%, 23.8%, 19.0%, 14.3%, 9.5%, and 4.8%. Cumulative distribution values are then calculated for the probability values. For the previous example, these values are as follows: 0, 0.286, 0.524, 0.714, 0.857, 0.952, and 1.000. Finally, a uniformly distributed random value (between 0 and 1) is generated to select the chromosomes to be mated. For example; in our previous example, if the generated random variable was 0.930, then the selected chromosome will be the forth one.

(4) Crossover process: each selected pair of chromosomes from the previous step will generate two offsprings. Each offspring shares genes from its parents. This is done by slicing the chromosomes and crossing over their genes. The crossover location is randomly generated to be uniformly distributed between 0 and N. An example of that is shown in Figure 1.1.

Figure 1-1. Cross-over example in Genetic Algorithm (GA)

(5) Mutation process: with a particular probability (relatively very small, typically 1 to 5%), there is a chance for each gene in the generated offspring to be swapped (perturbed) from a pool of all possible values. In the case of binary gene representation (Figure 1.1), a gene can be mutated by swapping its value from 1 to 0 or 0 to 1. In the case of non-binary gene

representation, a gene can be mutated by exchanging its value from a pool that contains all the possible values for a gene.

(6) Fitness evaluation for the generated offsprings: the fitness values for each of the *P* generated offspring solutions are found.

(7) Generate the new generation of solution: select the fittest *P* chromosomes from the parents and offsprings.

(8) If stopping criteria is achieved, stop; otherwise go to step 3.

Elitism (associated with Step 7) is usually used when running a GA. It means selecting the best chromosomes from the previous generation. This is because there is a chance that we might lose the best solution by not selecting it if we do not employ elitism. GA is usually used to solve unconstrained combinatorial optimization problems. Nevertheless, it can be used to solve constrained optimization problems. This can be achieved by different approaches. The approach we use in this work is to give a non-feasible solution a cost of *infinity* if the problem is a minimization problem. We also used GA to solve for continuous decision variables by creating a pool of values for each gene that contains a certain number of discrete values. For example, if we know that a decision variable lies between 0 and 1, then we can create a pool that contains: 0, 0.1, 0.2, …, 1. Some parameters that could considerably affect the accuracy and the execution time for a GA solution are: mutation rate, population size, cross-over rate. We use GA in this dissertation to solve the inspection planning problem in Chapters 3 and 7.

1.1.4 Simulated Annealing

Simulated Annealing is a metaheuristic used to solve combinatorial optimization problems. SA exploits the analogy of the annealing process in metallurgy in order to achieve the minimum energy crystalline structure. The fundamental idea is to escape local minima by accepting non-improving solutions at certain rate. We accept all improving solutions and accept non-improving

solution with a probability of $\min\left(1, e^{-\Delta T/t}\right).100\%$ in the case of minimization; where ΔT represents the difference between the current and the previous solutions and t represents the temperature. Notice if the current solution is better than the previous one ($\Delta T \leq 0$), we accept the current solution with 100% probability. The rate of accepting non-improving solutions is reduced with time according to the so-called *annealing* (*cooling*) *schedule*: $t_{i+1} = \alpha_r . t_i$ where α_r is the temperature reduction rate. There are different parameters that determine the quality of an SA solution such as: the annealing schedule (and initial temperature), initial solution generation (random or based on a heuristic), and the exchange strategy used to generate neighboring solutions. We use SA in Chapter 6 to maximize the rolled yield throughput.

1.1.5 Ant Colony Optimization

Ant Colony Optimization [developed by Dorigo (1996)] is a metaheuristic that was inspired by the social behavior of ants in finding the shortest paths from their nest (colony) to a food source. Although ants cannot see the *big picture* when they move over different paths, they usually end up following the shortest path by coordinating with each other through *depositing* and *sniffing* a chemical substance called *pheromone.* The deposited pheromone evaporates with time. Therefore, after few tours from and to the nest, more pheromone will be condensed in the shorter paths compared to the longer paths. This is because the deposited pheromone amounts evaporate faster in the longer paths. Consequently, more ants will be attracted to the shorter paths. ACO has been successfully used to solve combinatorial problems such as the Traveling Salesman Problem, Quadratic Assembly Problem and Routing Problem [Dorigo and Stützle (2004)]. To solve an optimization problem using ACO, the problem has to be represented by a connected graph (nodes and edges). Initially, we deposit a certain amount of pheromone in each edge in the graph after distributing a number of ants in nodes in the graph. After that, ants move from a node to another according to the *State Transition Rule*; which determines the probability of an ant to move from one node to another. The probability is determined by two values: pheromone amount (τ) and visibility (η). The visibility is usually determined by a greedy rule. For example; in the Traveling Salesman Problem, we use the greedy heuristic of moving to the next closest

city. Therefore, the visibility in that case will be the reciprocal of the distance between the current city and the other non-visited cities. After all the ants construct their tours, pheromone will be updated locally and globally according to the quality of the constructed tour solutions and the evaporation rate. A typical and general ACO algorithm can be summarized in Figure 1.2. There are different parameters that govern the performance of an ACO algorithm such as: number of ants, trail generation model, greedy heuristic to determine the visibility, evaporation rate, and the importance of pheromone versus visibility.

1. Initialize
   1.1. Represent the problem using a connected graph (nodes and edges)
   1.2. Distribute virtual ants (ants) in nodes (first node in the tour)
   1.3. Deposit a uniform amount pheromone in the edges
2. Construct ant tours using the *State Transition Rule*
3. Locally and Globally update pheromone amount in each edge
4. Stop if a stopping criterion is met; otherwise repeat from Step 2

Figure 1-2. Typical ACO algorithm

We use an ACO algorithm to solve the problem of DTM in Chapter 6.

1.1.6 Monte Carlo Simulation

Monte Carlo methods are numerical methods used to solve probabilistic and deterministic problems by taking samples from contributing populations and plugging them in the governing function of the system. Another definition is by Kalos and Whitlock (1986): "a numerical stochastic process; that is, it is a sequence of random events." Monte Carlo Simulation can be further explained as follows: given input random variables (X_1, X_2, …, X_N) with their probability distribution functions (PDF's) and the governing function that relates them with the output random variable $Y=f(X_1, X_2, \ldots, X_N)$, approximate behavior of the output random variable can be determined. After enough number of simulation iterations, distribution of the output random variable can be found. Apparently, increasing number of iterations increases the accuracy of the output. We use Monte Carlo Simulation in Chapters 3 and 7 to evaluate the objective function of the inspection planning problem.

1.1.7 Taguchi Loss Function

A comparison between two philosophies in quality engineering is pictorially explained in Figures 1.3 and 1.4. Conventionally, a quality characteristic used to be considered satisfactory so long as it lies between the lower and upper limits (*U* and *L*) in the step function (Figure 1.2). According to the step loss function; despite their closeness, *a* and *b* represent very different characteristics; *a* represents a reject-part (loss) and *b* represents a pass-part (no loss). In a further realistic recognition to customer satisfaction and strive for consistency, Genichi Taguchi developed more than 68 quality loss functions [Li (2002)] that quantify the loss function according to both customer and manufacturer perspectives. The most well-known one is the quadratic function that can be generally represented as follows for a single quality characteristic (*q*):

$$QL = M(q-t)^2 \qquad (1.3)$$

Where:

QL: Quality Loss; $\$$.

M: Constant; $\$/unit^2$.

q: Quality characteristic; *unit*.

t: Target value; *unit*.

Usually, quality characteristics behave randomly. Assuming a normal distribution behavior $q \sim N(\mu, \sigma^2)$, Equation 1.3 was proven to take the for shown in Equation 1.4 [Appendix E in Ross (1998)]. The first and second terms respectively represent the dispersions around the mean and the nominal values.

$$QL = M\left(\sigma^2 + \left(\mu - \frac{U+L}{2}\right)^2\right) \qquad (1.4)$$

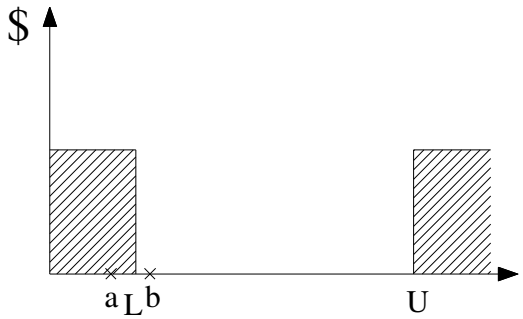


Figure 1-3. Step loss function

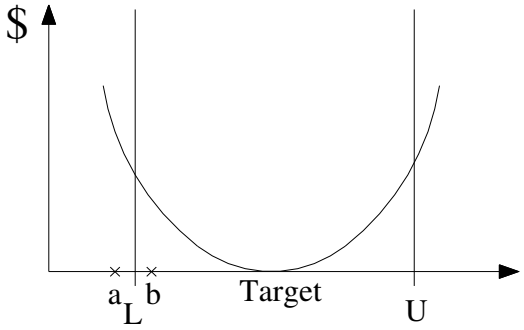


Figure 1-4. Taguchi loss function

It is unlikely to have only a single quality characteristic by which the quality is assessed. If the quality characteristics were not *correlated*, we can use the following *additive* function:

$$QL = \sum_{i=1}^{m} M_i \left(\sigma_i^{\,2} + \left(\mu_i - \frac{U_i + L_i}{2} \right)^2 \right) \qquad (1.5)$$

Where:

i: Quality characteristic index; $i = 1,...,m$

m: Number of quality characteristics.

However, if correlations between these quality characteristics cannot be ignored, then a different form for the quadratic loss functions has to be used. Although they did not consider these correlations in a model, [Byrne and Taguchi (1987)] described an example where two quality

characteristics (responses) interact. The example is about a tube that needs to be inserted into a connector. Functionality of this assembled part will require making the pull-off force as high as possible to keep the assembled part rigid during its operation; and the push force to be as low as possible to ease the assembly process. [Pignatiello (1993)] proposed the following general form for the correlated quality characteristics for a single part:

$$QL = (Q - \tau)^T C(Q - \tau) \qquad (1.6)$$

Where:

Q: Quality characteristic vector; $Q = (q_1, ..., q_m)$

τ : Target value vector; $\tau = (t_1, ..., t_m)$

C: Positive definite constant matrix. The diagonal elements for C represent the weights for the m quality characteristics and the off-diagonal represent the correlations. Notice that for $m = 1$, the loss function in Equation 1.6 leads to the form in Equation 1.2.

1.1.8 Additional Terms

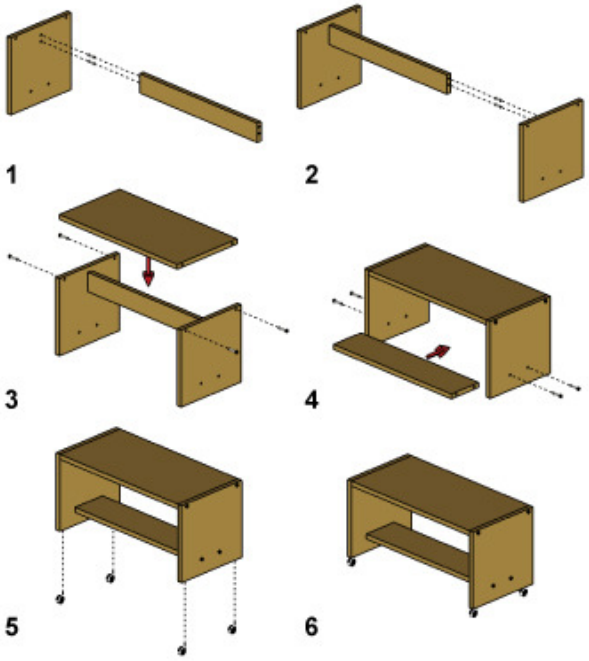


Figure 1-5: Step-by-step TV set table assembly [adapted from Agrawala et al. (2003)]

In the rest of this dissertation, we use the following terms: *subassembly group*, *subassembly item*, and *product*. Figures 1-5 and 1-6 show a simple assembly for a TV set table. For that assembly, we have five *subassembly groups* (Figure 1-6): (a) top shelf, (b) support board, (c) bottom shelf, (d) wheels, (e, f) side boards. A *subassembly item* will be for example one wheel from the wheels that we have. A product is the final assembly of the TV set table that is shown in 6 in Figure 1-6.

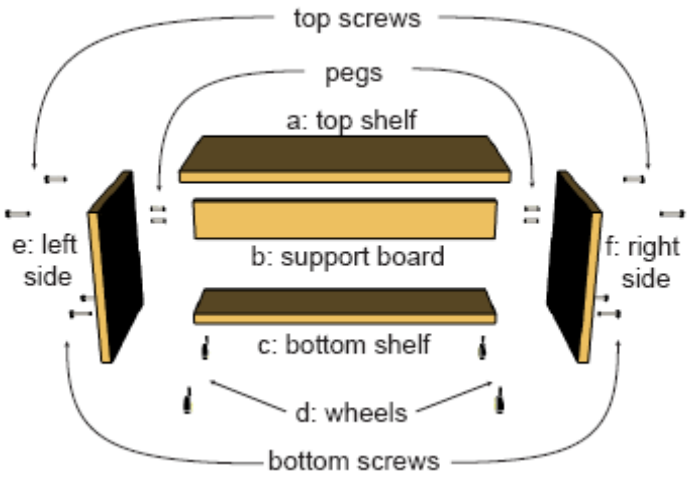


Figure 1-6: TV set table subassembly components (groups) [adapted from Agrawala et al. (2003)]

1.2. PROBLEM STATEMENT AND MOTIVATION

Increasingly confronted with global economy, agility in delivery, quality improvement, and cost effectiveness have become the three aspects determining the competitiveness and even the survival of a manufacturer. A successful manufacturing organization - in nowadays highly competitive manufacturing environment - is the one that implements agile production concepts, with cost minimization and variation reduction. In this dissertation, we present a comprehensive methodology to enhance the product competitiveness by achieving a good balance among the three conflicting factors. The main three parts in this work are explained next.

1.2.1. Strategic Variation Reduction Planning (SVR)

Inspection planning is to optimally determine what, when and how to inspect a feature/ process in an assembly (or machining) process. We aim to simultaneously achieve the cost minimization and variation reduction by minimizing the inspection, rework, scrap and failure costs of the assembly. *Inspection cost* refers to the cost of conducting inspection on a feature on a subassembly feature. *Rework cost* refers to the cost of converting (fixing) an out-of-tolerance feature to an acceptable range or to the nominal value. *Scrap cost* refers to the cost of scrapping a subassembly item if it is decided that it is more cost effective to do so. *Failure cost* is associated with the cost of discovering a failed assembly when the subassembly items are to be assembled. Intuitively, the costs of inspection, rework, and scrap are expected to be non-decreasing by increasing the frequency of inspection. On the other hand, the cost of failure for the final assembly is a non-increasing by increasing the frequency of inspection. The failure cost can be mapped with the other cost elements as we understand the relationship between the final assembly (final assembly quality characteristics; y) with the subassembly items (subassembly items quality characteristics; x's). This sort of mapping is necessary in order to estimate the total cost. Therefore, a set of historical data between the x's and y has to be available. However, for newly launched products or customized products, such data is usually not available. We propose simulating this data using some commercial software such as (DCS and VSA). In these software packages, the subassembly features (CAD features) are simulated according to given probability density function behaviors, and the behavior of the final assembly is generated accordingly. This topic is presented in Chapter 3.

1.2.2. Dynamic Variation Reduction (DVR), Enhanced Dynamic Variation Reduction (EDVR) and Dynamic Throughput Maximization (DTM)

In many industries, inspection data is known to merely serve for verification and validation purposes. They are rarely used to directly enhance the product quality because of the lack of approaches and difficulties of doing so. We try to directly improve the quality by exploiting the inspection data to reduce the variation of the final assemblies in *near-real-time* basis. We selectively mate subassembly items so we can reduce the variation of the final quality characteristics and/or increase the final rolled yield throughput of the final quality characteristics (defined in Section 1.1.2). We present three models and their solutions: Dynamic Variation

Reduction (DVR), Enhanced Dynamic Variation Reduction (EDVR) and Dynamic Throughput Maximization (DTM). The models have different objective functions and applications. The objective of DVR is to mate the subassembly items in order to minimize the drift between the nominal value and the final quality characteristics regardless of the final specifications (upper and lower limits). In EDVR, we add one more objective to the DVR model to make the objective function more analogous to Taguchi Loss Function (Section 1.1.7). The objective of EDVR is to mate the subassembly items in order to minimize the dispersions around the mean and the nominal values. This is also done regardless of the final customer requirements. On the other hand, the objective of DTM is to mate the subassembly items in order to maximize the rolled-yield-throughput of the final quality characteristics. These topics are presented in Chapters 4-6.

1.2.3. Hybrid Variation Reduction (HVR)

After inspecting a batch of subassemblies, we often have the choice of either assembling the inspected subassemblies arbitrarily or selectively. We propose integrating the inspection planning model with the dynamic variation reduction (throughput yield maximization) to further minimize the cost of failure of the final assemblies and therefore the total cost. This topic is presented in Chapter 7.

Figure 1.7 summarizes the framework for the proposed variation reduction techniques in this dissertation. The models and solution approaches for the strategic variation reduction (SVR) and dynamic variation reduction and throughput maximization (DVR, EDVR and DTM) are proposed independently in this dissertation. We use the SVR and DTM to further reduce the variation in Chapter 7. We can notice that DTM and EDVR are generated from the DVR model.

1.3. DISSERTATION ARCHITECTURE

This dissertation is divided into eight Chapters. The rest of the manuscript is organized as follows. A comprehensive literature review is presented in Chapter 2 for the following problems: variation reduction, inspection planning, selective assembly and sensor placement (allocation).

The strategic variation reduction problem and its solution approach are discussed in details in Chapter 3. The strategic variation reduction is based on developing an optimal inspection plans. It was found that the cost function cannot be optimized using a traditional mathematical programming approach, therefore we have resorted to a Genetic Algorithm. Inspection planning approach for newly launched products is also proposed in this Chapter. We also presented some numerical examples for illustrative and demonstrative purposes.

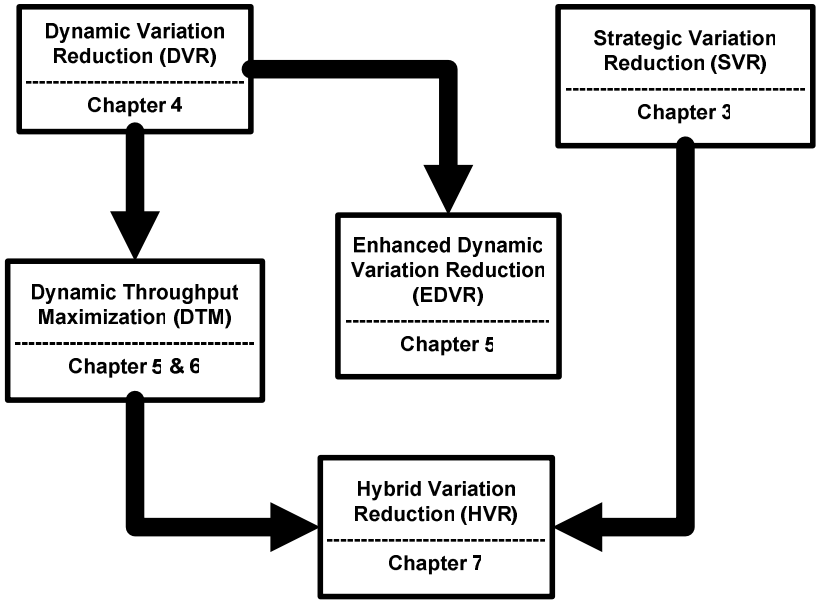


Figure 1-7. Framework of this dissertation

Chapters 4 and 5 discuss the Dynamic Variation Reduction (DVR), Enhanced Dynamic Variation Reduction (EDVR) and Dynamic Rolled Yield Throughput Maximization (DTM) problems and their solution approaches. The solution approaches are based on simple rules and metaheuristics (Simulated Annealing (SA) and Ant Colony Optimization (ACO)). These approaches are presented in Chapter 6. Numerical examples are solved for DVR and DTM in those Chapters.

An opportunity for further variation reduction is suggested by exploring both the strategic and dynamic variation reduction hybridly. This is detailed in Chapter 7 and supported with a

numerical example to verify the intuitive idea of implementing both the strategic and dynamic variation reduction techniques in the same time.

Finally, we conclude with summary, discussion and future research ideas in Chapter 8.

1.4. ORIGINALITY OF THIS WORK

The originality of this work is multifold. First, inspection planning for newly launched products is new in the literature. It is based on CAD data, frequency of inspection and general reaction plan. Second, the models and approaches to solve the dynamic variation reduction and throughput maximization are novel. Third, the integration of the strategic and dynamic variation reduction approaches to further reduce the variation has not been discussed in the manufacturing literature before. Forth, several metaheuristics (Genetic Algorithm, Simulated Annealing and Ant Colony Optimization) were used extensively in this dissertation and provided good results in solving complex combinatorial optimization problems. A new metaheuristics such as Ant Colony Optimization (ACO) has not been widely used in the manufacturing and production literatures.

CHAPTER 2: LITERATURE REVIEW

In this chapter, we present a literature review for the main topics addressed in this dissertation. These topics are: *inspection planning*, *sensor allocation*, and *selective assembly*.

2.1. INSPECTION PLANNING

Inspection planning can be defined as the act of determining *what*, *when* and *how* to inspect a feature in a product/ process. There are two types of inspection: product and process inspection. What we deal with in this work is the product inspection, which is the inspection of the quality characteristics (geometry and mechanical properties) of the products. From this point on, we will refer to the product inspection planning as *inspection planning*. The process inspection (sometimes referred to as the *sensor allocation problem*) is the inspection of the manufacturing process equipment such as the fixture, cutting tool wear, etc. In the following two Sections (Sections 2.1.1 and 2.1.2), we provide a recent literature survey for the two problems.

2.1.1. Process Inspection Planning (Sensor Allocation Problem)

Sensor allocation problem is the opposite of the diagnosis problem. The diagnosis problem can be defined as follows: given a set of symptoms (inspection information), what are the most likely *disorders* (e.g. diseases, production failure, etc) to take place? The sensor allocation problem can be described as follows: given a set of *disorders* (e.g. extremely high variation), what are the most significant features to be inspected that will maximize the diagnosibility. Azam et al. (2004) modeled the sensor allocation problem as a Knapsack problem to maximize the *diagnosibility*. Lindsay and Bishop (1964) used dynamic programming. Ding et al. (2003) modeled it using the Stream of Variation Approach (SOVA). Khan et al. (1998) focused their attention on the problem of allocating three sensors on a fixture unit that implements the 3-2-1 principle. Binglin et al. (1993) used a Genetic Algorithm to solve the diagnosis problem by maximizing a likelihood function for the *diagnosibility*.

2.1.2. Product Inspection Planning

Chen and Thornton (1999) used Monte Carlo Simulation and a Simulated Annealing algorithm (SA) to solve the inspection planning problem for the parts. Our work was initially inspired by their work. More discussion about their work is elaborated later in Chapter 3. Greenstein and Rabinowitz (1997) solved the problem statistically in two stages. The objective in their study was to fully inspect $K<n$ components in the first stage that "explain" the whole behavior of the n components. Their objective function is to minimize the cost of accepting a "bad" product, the cost of rejecting a "good product" and the cost of inspection. After that, they determine whether it is cheaper to inspect the rest of the batch or not. The authors assumed that the joint probability distribution function of the components is known and that it is normally distributed. Moreover, they did not consider any possibility for rework or scrap actions in their model and the specification limits were input information rather than being decision variables. Chen and Chung (1996) introduced a model to determine the inspection precision and the optimal number of repeated measurements in order to maximize the net expected profit per item. The model is specifically applicable for the lower specification-limit quality characteristic; i.e. the specification has unbounded upper limit. The profit is modeled as the difference between the selling price and the following costs: inspection, production, and dissatisfaction of the customer. There is an assumption that all measurements are normally distributed and all items are completely inspected at least once because of inspection inaccuracy. Their model is mostly appropriate for industries where there is a need for repeated measurements because of known measurement errors and where the production is at late a stage of producing an item in the supply chain.

2.2. SELECTIVE ASSEMBLY

Selective assembly is the act of matching *groups* of subassembly items in order to reduce the final variation. It has been used when subassembly variations are high so if assembled arbitrarily, design and customer requirements will not be satisfied. We have realized that since we are

planning for inspection planning as another technique to reduce variation, the collected inspection data could be utilized to mate subassembly items with each other in order to produce the least possible variation. Mansoor (1961) concluded that *mismatching* in the number of items in groups is a serious problem in selective assembly. Therefore, the author classified the selective assembly problem according to the relationship of the natural process tolerances to the tolerance of the fit. Desmond and Setty (1962) presented a simple approach for mating subassembly groups. Pugh (1992) dealt with selective assembly for two parts when the subassemblies have dissimilar variances. The author suggested truncating the subassembly with the larger variance. Also, Pugh (1986) developed a code that helps engineers to develop groups and then assign them. More recently, Kannan et al. (2003) used a Genetic Algorithm to solve the problem by generating six partitions for each component (subassembly). Whitney (2004) discussed three strategies for reducing variation: selective assembly, Build to Print and Functional Build. The author provided interesting comments on when selective assembly could have difficulty without providing a mathematical model and a procedure for solving the problem.

CHAPTER 3: STRATEGIC VARIATION REDUCTION (SVR)

3.1. INTRODUCTION

Thousands of new products are produced every year and due to harsh competition there is no more enough time to iterate with experiments until the product is ready to be sent to the market. Therefore; agility, cost minimization and quality near-perfection have become the backbone of any successful manufacturer.

Assume you were hired to rescue a traditional manufacturing facility that is almost going out of business despite the fact that you have intelligent employees and innovative designs. Also, let us assume that the resources you have are very limited so you cannot acquire new equipment and machines. Nonetheless, you are now facing the challenge of managing a facility that is aimed to be capable of producing new products that have never been manufactured before and there is scare or even no knowledge about the products. These products need to be marketable as soon as possible and their functions and performances must be consistent and satisfactory to the customer requirements. Several solutions could be proposed in this situation. One of them is to develop proactive and strategic inspection plans so all the goals and constraints are entirely fulfilled.

This Chapter is divided into five Sections and organized as follows: in Section 3.2, we define the problem and introduce the cost function we work on optimizing. Next, we present our mathematical model and our methodology in Section 3.3. We also raise the question whether the frequency of inspection has to be a decision variable in minimizing the total cost. Numerical examples are solved afterwards in Section 3.4 for illustrative purposes for our findings. Finally, we close this Chapter with concluding and summary remarks in Sections 3.5.

3.2. PROBLEM

We can define inspection planning as the act of identifying what, when (how often), how to measure subassemblies and the corrective action based on the inspection. In addition to the inspection planning for existing products, we solve the challenging problem of inspection planning for newly launched products in this Chapter. This problem is rather complicated because of the lack of historical data that maps the relationships between the subassemblies and the final assemblies especially for complex products. Inspection planning is considered a strategic and static variation reduction technique because it is designed for a particular production system throughout its lifecycle. This does not mean that there is no need for revising the inspection plan every now and then.

We do not consider in our work here the sensor allocation problem that aims at reducing product variation by inspecting the processing devices and machine components that make the parts, such as fixtures, cutting tools, etc. Rather, we look into what subassembly parts need to be inspected and what to do if the inspected feature was found to be out of tolerance in order to minimize the total cost of inspection, rework, scrap and failure. In this work, we use the objective function (Equation 3.1) that was proposed in Chen and Thornton (1999). It is the total cost of inspection (C_I), scrap (C_S), rework (C_R) and failure (C_F). It is essential to mention at this point that the inspection, scrapping and rework costs are all associated with the subassembly quality characteristics, where the failure cost is associated with the quality characteristics of the final assembly. It is intuitive that when the failure cost is high, it is generally better to inspect subassemblies. In contrast, it is generally better (more cost-effective) not to inspect when the failure cost is negligible. We will elaborate on that later in the Chapter.

$$\text{Minimize } TC = C_I + C_S + C_R + C_F \qquad (3.1)$$

We assume that we are dealing with simple assemblies that are made of only one level of subassemblies. This means that the final assemblies are made of subassemblies that stand-alone. It is assumed not to have subassemblies of subassemblies. We also assume that a final assembly is made of M subassemblies and that each subassembly contains a single quality characteristic. The final assembly contains N quality characteristics.

Assuming normality for the contributing (subassembly) quality characteristics and concluding (final assembly) quality characteristics, the objective function in Equation 3.1 can be further decomposed as shown in Equation 3.2.

Minimize

$$TC = R\sum_{t=1}^{M} freq_t \left\{ c_{It} + c_{Rt} P_{Rt}\left(LL_{S_t}, LL_{R_t}, UL_{S_t}, UL_{R_t}, \mu_t, \sigma_t\right) + c_{St} P_{St}\left(LL_{S_t}, UL_{S_t}, \mu_t, \sigma_t\right)\right\} + Q\sum_{a=1}^{N} c_{Fa} P_{Fa}\left(LL_a, UL_a, \mu_a, \sigma_a\right) \quad (3.2)$$

Where:

TC: Total cost

R: The number of items in each subassembly group before inspection

$freq_t$: Frequency of inspecting subassembly quality characteristic *t*

c_{It}: Inspection cost per subassembly quality characteristic *t*

c_{Rt}: Rework cost per subassembly quality characteristic *t*

P_{Rt}: Probability of reworking quality characteristic *t* (area under II and III in Figure 3.1)

c_{St}: Scrap cost per subassembly for quality characteristic *t*

P_{St}: Probability of scrapping for quality characteristic *t* (area under I and IV in Figure 3.1)

c_{Fa}: Failure cost per final assembly quality characteristic *a*

P_{Fa}: Probability of failure for the final assembly quality characteristic *a*

M: Number of subassembly groups

N: Number of final assembly quality characteristics

Q: Maximum number of items in a subassembly group to be assembled, $Q \le R$ because the scrapped items will not be used for final assembly.

LL_{Rt}, LL_{St}: Lower limits associated with reworking and scrapping the t^{th} inspected subassembly items; respectively (Figure 3.1).

UL_R , UL_{St}: Upper limits associated with reworking and scrapping the t^{th} inspected subassembly item; respectively (Figure 3.1).

The quality characteristic of the final assembly (a) is assumed to be normally distributed; such that:

$$q_a \sim N\left(\mu_a, {\sigma_a}^2\right)$$

$$q_a = f\left(freq_1,..., freq_M, \mu_1,..., \mu_M, \sigma_1,..., \sigma_M, LL_{S_1},..., LL_{S_M}, LL_{R_1},..., LL_{R_M}, UL_{S_1},..., UL_{S_M}, UL_{R_1},..., UL_{R_M}\right)$$

The previous equation states that a final quality characteristic a is distributed according to the parameters and variables inside the bracket. The mapping between the subassembly quality characteristics and the final assembly quality characteristics is not usually known unless there is data for an existing product. This will make it hard to estimate P_{FA} (failure probability of final quality characteristics) for newly launched products because the probability density function of the final characteristics (q_a) is not known for a given subassembly quality characteristics.

The cost function in Equation 3.2 cannot be easily dealt with analytically because there are no explicit forms to express the probabilities of rework, scrap and failure (P_{Rt},P_{St}, P_{Fa}). Although it is possible and yet tedious to fit functions that give probabilities (P_{Rt},P_{St}) for the given inputs (*UL*'s and *LL*'s), it is not reasonable to do so unless the functions are normally distributed and the functions that map the contributing quality characteristics to the final assembly quality characteristics are known. Further, it is also not easy to estimate P_{Fa} because the probability density function of a is not usually known as the mapping function is unknown. Therefore, we resort to Monte Carlo simulation to generalize the model as it was proposed by Chen and Thornton (1999). Notice that we have introduced the *frequency* of inspecting a quality characteristic for a subassembly t ($freq_t$) as a portion of the objective function. Later in this Chapter, we will discuss whether the frequency of inspection can be a binary or continuous decision variable. Moreover, we introduce two ranges of a tolerance (LL_S, LL_R, UL_S, UL_R) for each quality characteristic t. The reason for that is to generalize the action of rework and scrap compared to what is found in Chen and Thornton (1999) where they proposed that a subassembly part has to be either reworked or scrapped if it is out-of-tolerance regardless of its measured value. What was proposed can be enhanced by introducing two additional decision variables for each subassembly t. Referring to Figure 3.1, when a subassembly item t is found to be in regions I or IV (so far from the nominal value), then we scrap it. However, if it is in regions II and III (not so far from the nominal value), then we rework it. Finally; if otherwise, we keep it.

This is how we propose implementing a more realistic corrective plan. It is practical to impose some functional constraints sometimes. If we know for a fact that it is impossible to rework an inspected subassembly item if a dimension is less or larger than a specific value, then we can impose a constraint that allows no rework: $LL_S=LL_R$ and $UL_R=UL_S$.

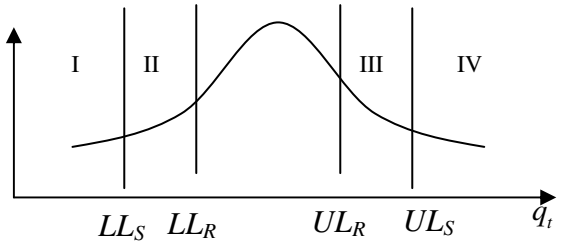


Figure 3-1. Quality characteristic for subassembly item *t*, I & IV: scrap areas; II & III: rework areas

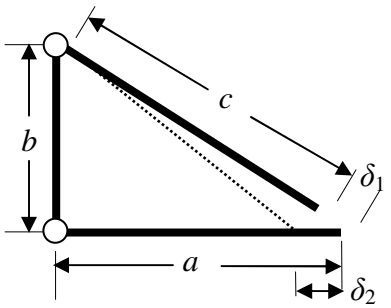


Figure 3-2. Simple nonlinear assembly

The number of decision variables for our problem here is 5.*M*; where *M* is the number of subassembly groups. The five decision variables for a single subassembly are: $freq_t$, LL_S, LL_R, UL_R, UL_S. Suppose we have three quality characteristics that are associated with three subassemblies (*a*, *b* and *c* dimensions) that comprise final assembly requirements δ_1 and δ_2. Example for this case can be shown in Figure 3.2. In this case, we are interested in mapping the relationship between *a*, *b* and *c* with δ_1 and δ_2. Sometimes, these mapping functions can be easily found as in the case of our example in Figure 3.2. The mapping functions are shown in Equation 3.3. However, most of the time, it is hard to map these quality characteristics, therefore as a main part of our methodology; we are proposing using simulated-CAD data (DCS or VSA) and then

regression to evaluate the mapping functions. This is discussed in more details in the next Section.

$$\begin{aligned} \delta &= f(a,b,c) \\ \delta_1 &= f(a,b,c) = c - \sqrt{a^2 + b^2} \\ \delta_2 &= f(a,b,c) = a - \sqrt{c^2 - b^2} \end{aligned} \qquad (3.3)$$

For the example shown in Figure 3.2, the inputs and the decision variables to conduct inspection planning search are:

Inputs (Parameters): $UL_{\delta 1}$, $UL_{\delta 2}$, $LL_{\delta 1}$, $LL_{\delta 2}$, PDF_a, PDF_b, PDF_c, c_I, c_R, c_F, c_S
Outputs (Decision variables): LLS_a, LLR_a, ULR_a, ULS_a, LLS_b, LLR_b, ULR_b, ULS_b, LLS_c, LLR_c, ULR_c, ULS_a, $freq_a$,$freq_b$,$freq_c$.

3.3. Approach

In order to find the failure cost in Equation 3.2 ($c_{FA.}P_{FA}$), we need to find a function that maps the input variation data to the outputs. Therefore, for new products we may not know the function that maps the subassembly quality characteristics with the final assembly quality characteristics because of the lack of historical data and/or the complexity of the assembly. We propose using CAD-variation analysis software such as 3DCS or VSA to find out that function (refer to Figure 3.3). 3DCS and VSA are CAD-Monte Carlo simulation based software that analyzes the tolerance stackup for an assembly. What these software packages do is simulating CAD features for subassembly items for a number of replications according to given probability density function behaviors and then predict the behavior of associated the final quality characteristics.

Since we cannot generally express the objective function analytically as we discussed earlier, we propose using a Genetic Algorithm (GA) to search for the optimal solution. Our approach using a GA is summarized in a flowchart in Figure 3.5. Figure 3.6 further decomposes the cost estimation functions in the right hand half of Figure 3.5.

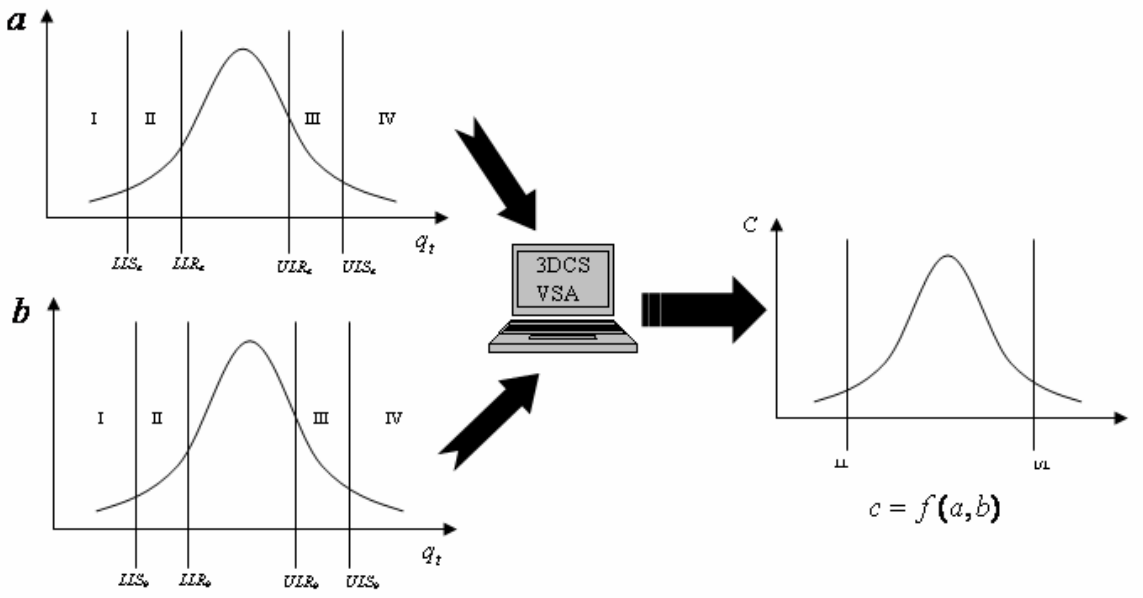


Figure 3-3. Resultant variation data for the final assembly *c*. Note: the function that maps the input variation data (*a* and *b*) with the output variation data is evaluated by regression when the data are obtained from 3DCS or VSA.

With *frequency of inspection* for the *M* subassembly quality characteristics as the only decision variables, all the terms in the objective function in Equation 3.2 are linear as shown in Figure 3.4 if all the quality characteristics behave according to normal distributions. Hence, the total objective function will be linear. This means that the optimal inspection plan will be to either to fully inspect a subassembly (feature) *t* or not to inspect it at all; i.e. $freq_t$=1 or 0. Refer to Figure 3.4, we can say that the inspection, rework and scrap costs increase by increasing the inspection intensity. On the other hand, increasing the inspection intensity decreases the failure cost. Therefore, our inspection plan has to be a trade-off between those costs. This suggests that if the failure cost is the dominant cost (i.e. $C_F >> C_I+C_R+C_S$), then the final cost function will be non-decreasing. Therefore, it would be cheaper to fully inspect. This is the case if we have an unconstrained objective function. We will show later that we will be better off if we partially inspect a subassembly item if we consider a minimum yield as a constraint.

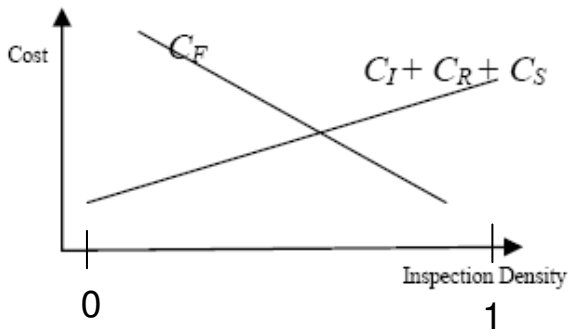


Figure 3-4. Unconstrained cost function behaviors (no yield constraint)

3.3.1. Genetic Algorithm (GA)

What you see in Figure 3.5 is the same procedure that we introduced here for GA in Chapter 1 but with the following modifications:

1. Since the decision variables are continuous, we created a *discrete pool* of gene values from which a value can be selected. For instance, *frequency* can take a value between 0 and 1. Hence, we created a function (its name is *Discritizer* function in the flowchart) that can match a randomly generated variable between 0 and 1 with the pool of gene values that we selected. In this Chapter, we generated three pools (sets): (1) 0 or 1, (2) [0.0, 0.1, 0.2… 1.0], and (3) [0.00, 0.05, 0.10, 0.15… 1.00]. Apparently, finer pools (sets) can generate more accurate solution but they are more computationally intensive. On the other hand, we discritized the specification limits differently (UL_R, UL_S, LL_R, LL_S). These decision variables were given sigma numbers; i.e. UL_{Rt}=3 means $\mu_t+3\sigma_t$ value for a quality characteristic t.

2. For each quality characteristic, t; in order for a string (chromosome) to be feasible, we forced the constraint that $UL_R \geq UL_S$ and $LL_s \geq LL_R$.

3. We impose the constraint of minimum yield (as it is shown in Figure 3.6) by giving the total cost *infinity* value when the minimum yield constraint is not satisfied.

3.3.2. Cost Estimation

Figure 3.6 summarizes how we estimate the cost (fitness) by using Monte Carlo Simulation. This is a modified approach to what Chen and Thornton (1999) proposed. The thick arrows in the Figure connect the external data (such as CAD data, Chromosome, etc) to the internal functions. The flowchart is divided into two components: virtual inspection and virtual assembly. It starts off through design decision for all possible subassembly quality characteristic (*X*'s) contributors on the final requirements (*Y*'s) and by assuming the behavior of those chosen *X*'s (upper left corner of Figure 3.5). After that, enough simulations are run through 3DCS or VSA to find the *Y*'s that are associated with the generated *X*'s. The *Y*'s are then mapped with the *X*'s through regression. This fitted function will be needed in the final assembly part of the simulation.

To find a cost, we generate a random variable that represents the t^{th} subassembly according to the known *PDF* behavior; X_{tr}. We put the generated quality characteristic into the inspection process according to a given frequency. If the generated value (X_{tr}) was found to be in region I or IV (Figure 3.1), then we scrap it (send nothing to be assembled by giving the X_{tr} a value of zero) and we add that cost of scrapping. However, if it is located in II or III, then we rework it to the near nominal value and we add that cost. If the part was not inspected, then we send it right away to the final assembly batch. Only the non-inspected and the reworked parts are sent to the final assembly batch. After checking all the input subassembly quality characteristics (t=1,…,M) for all the given parts (r=1,…,R), we reach to a point where we have different number of subassemblies because of the scrap procedure. Suppose there are 2 subassembly groups (*a* and *b*) of size 100 for each one of them. If 50/0 parts of subassembly groups *a*/*b* were scrapped, this means that only 50% of subassembly *b* will be utilized and 100% of the resultant *a*'s will be utilized. At that point, we can say that the yield reduced from 100% to 50% because of the inspection. The number of parts for this example will be 50 (Q=50 in the final inspection simulation). Q can be calculated as follows:

$$Q = \min\left(\sum_{r=1}^{R} step(X'_{tr}), \forall t = 1,...,M\right)$$
$$step(y) = \begin{cases} 1, y > 0 \\ 0, otherwise \end{cases} \qquad (3.4)$$

In the virtual assembly part of the simulation (VA), we take a set of input *X*'s and find the associated *X*'s according to the function found after fitting the CAD data. Alternatively, we can feed the *X*'s data we collected to the CAD model to predict the *y*. If the part (*Y*) is found to be within specifications (*LL* and *UL*), then we proceed with the next part. However, if it does not fall within the specification limits, then we declare it as a failed assembly and we add the failure cost. After examining all the *Q* parts, we estimate the yield based on that. We notice that the final yield is dependent on the scrap and failure rates, as follows:

$$YRT = \frac{Q - Q_F}{R} \qquad (3.5)$$

Where:

YRT : Rolled Yield Throughput

Q : Maximum number of items in a subassembly to be assembled

R : Number of subassembly parts before inspection

Q_F : Number of failed subassembly parts after the final assembly, $Q_F = iCount_{Failed}$ (Figure 3.6)

In order to impose the yield constraint, if the found yield was below the minimum yield then we give the total cost a value of *infinity* so the solution can be excluded later in the GA.

3.3.3. $freq_t$: Is it a discrete or continuous decision variable?

Up to this point, we have found that the frequency of inspecting a subassembly *t* has to be always 0 or 1 when the objective function is unconstrained and the subassembly quality characteristic is normally distributed (Figure 3.4); which means it is optimal to fully inspect a quality characteristic or not to measure it at all. In the following remarks, we summarize our findings in regard of the frequency as a decision variable.

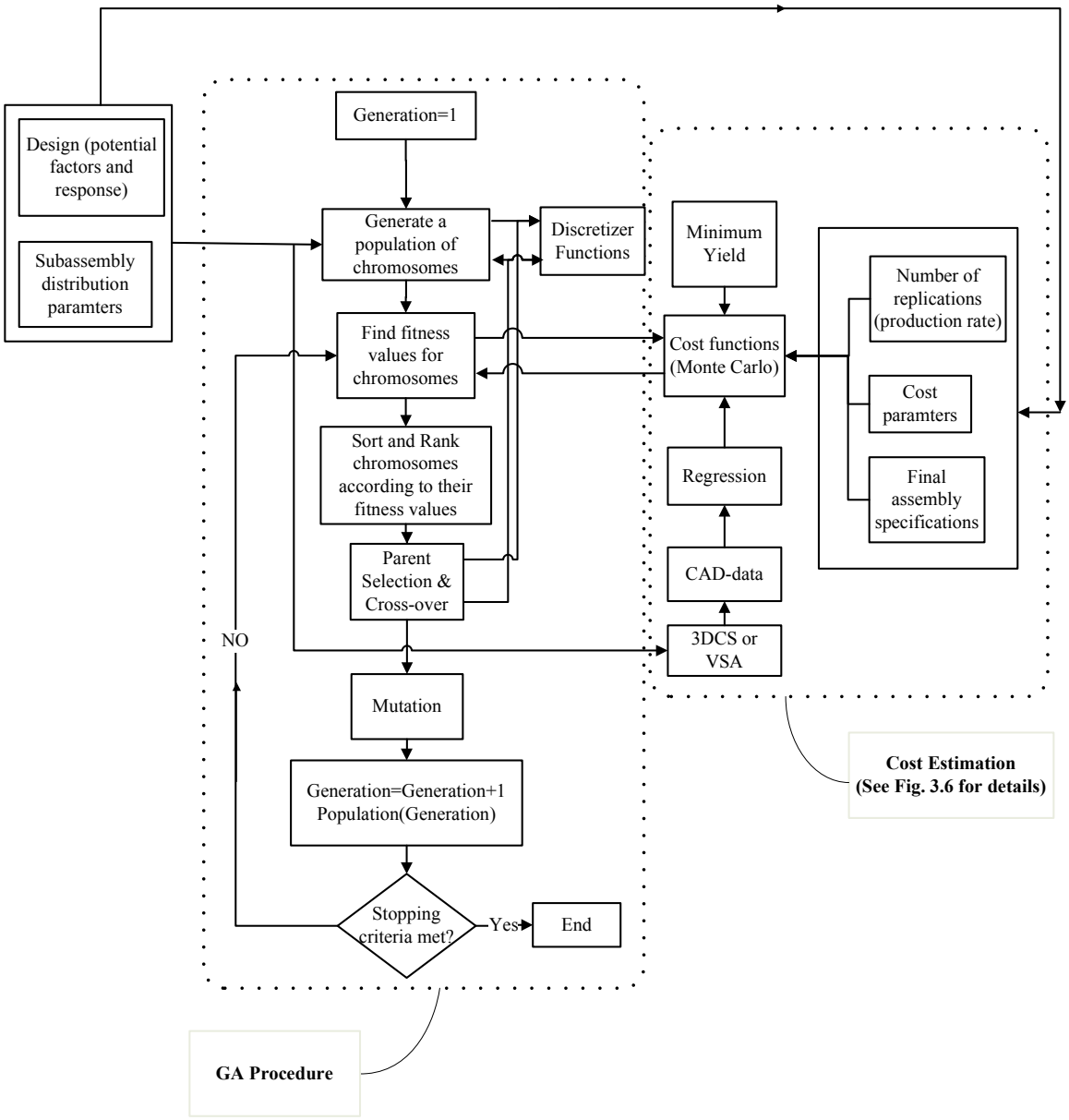


Figure 3-5. New product inspection planning flowchart using a Genetic Algorithm (GA)

Remark 3.1

When all the decision variables other than *frequency* are known, for a subassembly $t \in M$ that is normally distributed:

(1) When the effect of failure cost is more dominant than the other costs (inspection, rework and scrap), the optimal plan will be to fully inspect the subassembly t (i.e. $freq_t$=1).

(2) When the effect of failure cost is equal to the effect of other cost items, then fully, partially inspecting a subassembly or not inspecting it at all (inspecting it at any frequency between 0 and 1) will have a similar total cost.

(3) When the effect of inspection, rework and scrap cost is more dominant than the failure cost:

3.1. If the yield constraint $(yield \geq y_{\min})$ is not active, the optimal plan will be to not to inspect the subassembly t (i.e. $freq_t$=0).

3.2. If the yield constraint $(yield \geq y_{\min})$ is active, the optimal plan will be to *partially* or completely inspect the subassembly t ($freq_t$>0).

3.4. NUMERICAL EXAMPLES

In this Section, we solve two numerical examples using our proposed approach. We want to develop inspection plans for a product that has four input quality characteristics (X's, M=4) and three final quality characteristics (Y's, N=3). The behavior of Y's when X's change can be mapped by simulating the product using 3DCS or VSA. After that, we can relate the X's with the Y's by using regression. The objective is to determine the optimal frequencies and action specifications (rework and scrap limits) that give the minimum total cost. We also consider that we are constrained with a minimum yield. We did not run the CAD simulation package because it is not capable in providing the Y's that are associated with the generated X's. Therefore, we generated the X's to be all U(0,1). The Y's were generated according to the following functions:

$$y_1 = 100 - 50x_1 + x_2 - x_3 + 50x_4$$

$$y_2 = 35 - x_1 + x_2 - x_3 + 12x_4$$

$$y_3 = 2.5 - x_1 + x_2 + 2x_4 .$$

Example 1: Final Assembly Specifications: Lower Limits for the N quality characteristics are respectively: 80, 35 and 3.5 and the Upper Limits are respectively: 120, 42, and 5. The number of replications R=1000 (Initial number of items in a subassembly), Minimum Yield = 45%,

Maximum Number of GA Replication = 1000, c_I = \$10 (cost of inspecting a single quality characteristic); c_R = \$30 (cost of reworking a single quality characteristic); c_S = \$30 (cost of scrapping a single quality characteristic); c_F = \$0 (cost of failing a single final assembly), Population Size: 6 Chromosomes, Mutation Rate: 5%.

Table 3.1 shows the solutions when we consider different discrete frequency sets. We know from our experience with the data that the fourth quality characteristic has more significance than the other quality characteristics. This makes it intuitive to inspect that subassembly more often than the others as you can see in the Table. The Table suggests that by refining the resolution of the frequency set (pool), we will get less total cost because the frequency is a continuous decision variable in this case. Notice that the optimal frequency for the fourth quality characteristics is 40%, which did not change when we further refined the frequency set from set 2 to 3. This is also in agreement with the remark because the failure cost here is much less than the other cost parameters. The other optimal decision variables in that case are shown in Table 3.2. Figures 3.7 to 3.9 show the solution (maximum, minimum, and average objective function for GA population) over the progress of GA replications for the three sets. As part of our approach, we imposed the minimum throughput constraint by giving a plan that leads to less than the specified throughput a cost of *infinity*. The discrete behavior of the cost functions in Figures 3.7 to 3.9 is because we have those infinity costs that are not plotted.

Table 3.1. Inspection plans at several frequency resolutions, Example 1

Set	Frequency Resolution	$freq^*_1$	$freq^*_2$	$freq^*_3$	$freq^*_4$	Total Cost \$	Exec. Time [min]
1	{0,1}	0	0	0	1	16,300	26
2	{0,0.1,0.2…0.9,1}	0	0	0	0.4	8,330	26
3	{0,0.05,0.1…0.95,1}	0	0	0	0.4	8,210	26

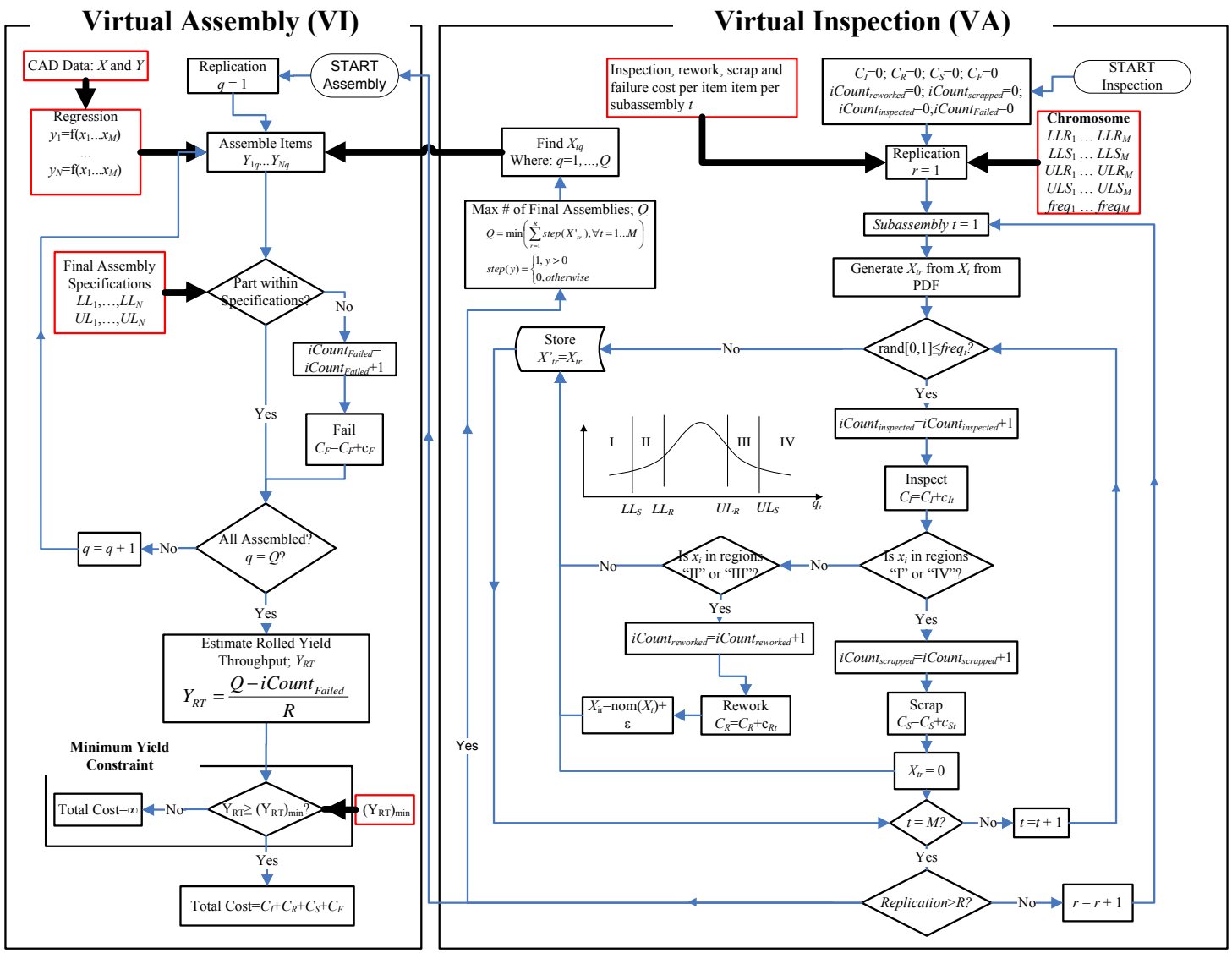


Figure 3-6. Cost estimation using Monte Carlo Simulation

Table 3.2. Example 1 decision variables found using our GA algorithm

Resolution	Decision Variables	X_1	X_2	X_3	X_4	Total Cost; $
1: {0,1}	*LLR**	3.5	1	3	3.5	16,300
	*LLS**	3.5	4.5	5	4.5	
	*ULR**	1	1	1	1	
	*ULS**	5	2	3	3.5	
	*freq**	0	0	0	1	
2: {0,0.1,0.2…0.9,1}	*LLR**	2.5	4	0.5	4.5	8,330
	*LLS**	4.5	5	3	4.5	
	*ULR**	5	1.5	1	0.5	
	*ULS**	5	3	5	3	
	*freq**	0	0	0	0.4	
3: {0,0.05,0.1…0.95,1}	*LLR**	3.5	0.5	2	2	8,120
	*LLS**	5	2	3.5	5	
	*ULR**	1.5	2.5	2	0.5	
	*ULS**	3.5	3	3.5	3	
	*freq**	0	0	0	0.4	

Example 2: The data used here are the same as the previous example except for the cost failure cost parameter and the minimum throughput requirement. In this example, we assume the failure cost to be $1 per failed final assembly; i.e. c_F = $1. We also examine the effect of changing the required throughput on the total cost and the required frequency of inspection of the forth subassembly. The results of the simulation are shown in Figures 3.10 and 3.11. Notice that when we changed the minimum required throughput from 0 to 40%, the cost was found to be constant ($630) and the optimal frequency for inspection of the forth subassembly was found to be 0. This behavior changes when we increase the throughput from 45% to 70%, where we see an increase in both the frequency of inspection and the total cost. We found that imposing a throughput higher than 70% would never lead to a feasible solution. Therefore, we do not show any results after a throughput of 70%. Moreover, we can notice that the optimal frequency of inspection and total cost changed considerably by increasing the failure cost from $0 to $1. The optimal frequency increased from 40% to 45% and the optimal cost increased from $8,120 to $9,300.

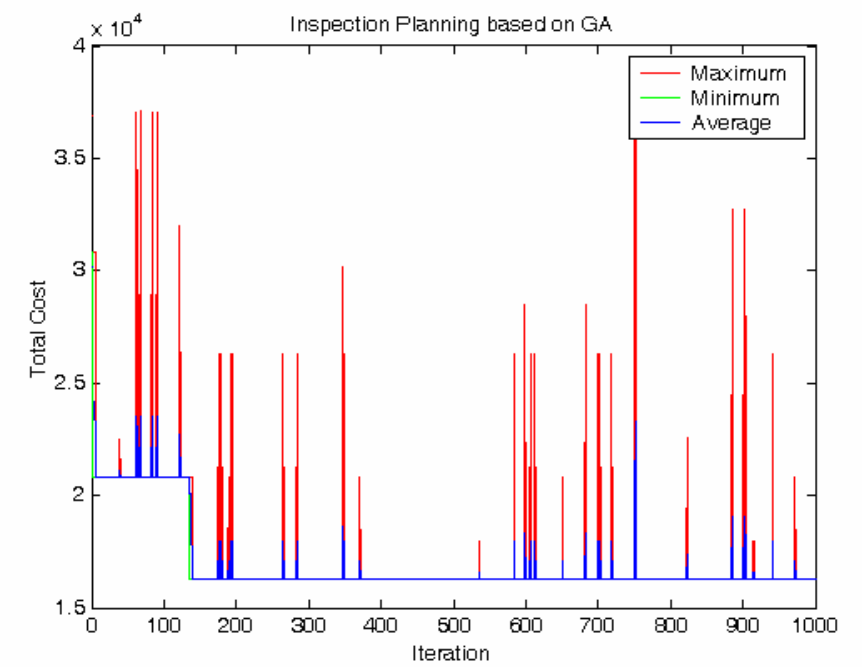


Figure 3-7. Set 1 solution in Example 1 (C_F=\$0)

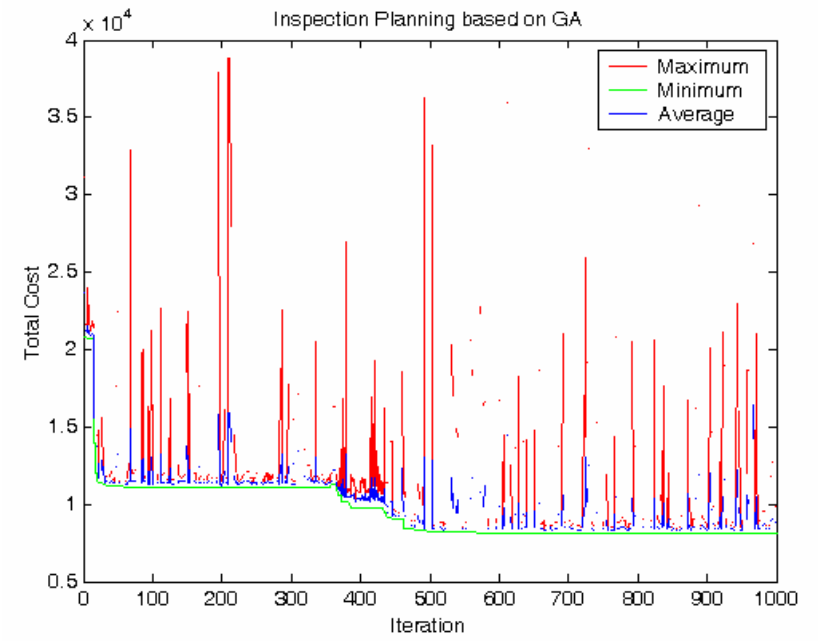


Figure 3-8. Set 2 solution in Example 1 (C_F =\$0)

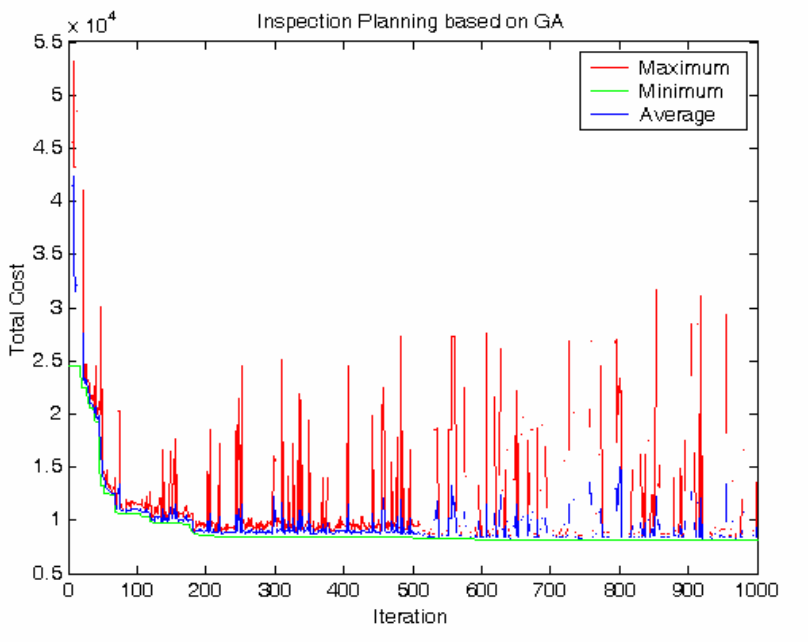


Figure 3-9. Set 3 solution in Example 1 (C_F =\$0)

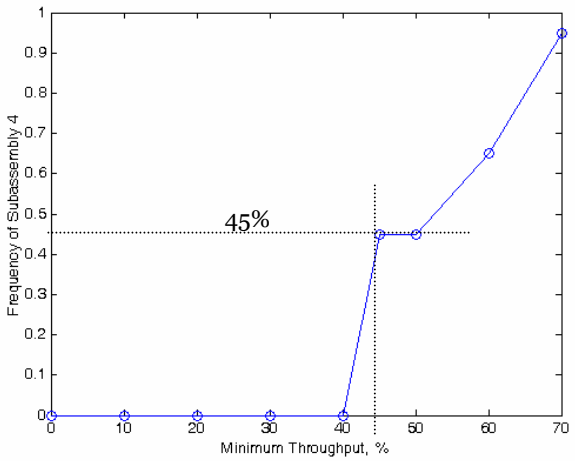


Figure 3-10. Effect of minimum yield on the optimal frequency of inspection for the forth subassembly (C_F=\$1); Example 2

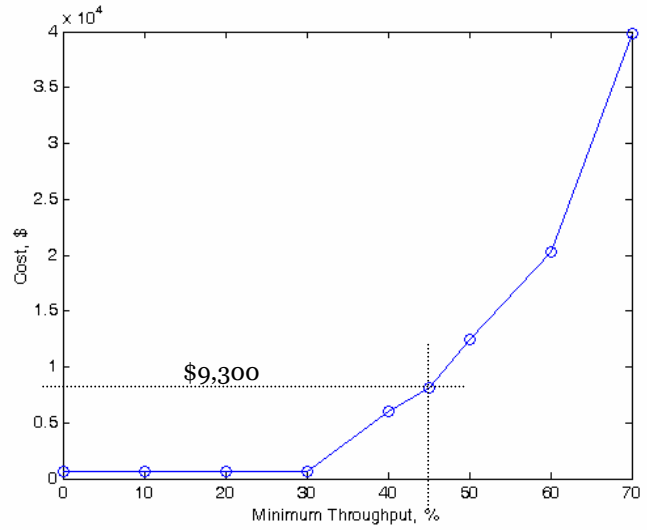


Figure 3-11. Effect of minimum throughput on the final cost (C_F=$1); Example 2

3.5. SUMMARY

In summary, we proposed an approach based on simulated data that we can obtain from CAD Variation Prediction packages (3DCS and VSA) and Genetic Algorithm. The approach is mainly designed to develop inspection plans for newly launched products. Moreover, we proposed a general and realistic action approach by introducing two tolerance limits with which keeping, reworking and scrapping an item are all possible. Frequency of inspection was also introduced as a continuous decision variable that can further minimize the total cost. We summarize our conclusions and findings as follows:

(1) We can notice that by refining the resolution of the frequency set (Table 3.1), the found objective function is found to decrease because the optimal frequency values can be more accurately determined. This indicates that the frequency of inspection is a valid decision variable. In that case, the cost decreased 50% by refining the frequency resolution from {0, 1} to {0, 0.1, 0.2, …, 1}.

(2) We also verified our remark 3.1 in this Chapter by solving the numerical examples. In the examples, the failure cost was negligible compared to the other cost factors. It was found that it

is more cost-effective to partially inspect the forth subassembly group rather than fully inspecting it.

(3) We can see from Figures 3.10 and 3.11 in Example 2 that increasing the required minimum yield increases the optimal frequency of inspection and the total cost.

CHAPTER 4: DYNAMIC VARIATION REDUCTION (DVR)

4.1. INTRODUCTION

Assemblers are often left with many subassemblies that have been inspected and found to have different quality characteristic values. If done at all, matching (mating) subassemblies together to result in the best final assemblies may be both awkward and hectic. We propose a mathematical model in this Chapter and new approaches that can match subassembly parts together in order to get the minimum possible variation of producing a batch in an assembly line. We propose a heuristic that is proven to be optimal for a simple case. We also verified that we can reach a near-optimal solution for more complicated cases by comparing the solution with a benchmark solution using Genetic Algorithm. It was found that the most advantage can be achieved from our approach is when the subassembly variations are close. Possibility for implementing our approach for selective assembly is also discussed.

In our work, we do not assign groups of subassemblies together (selective assembly); rather we assign individual subassemblies together. No mathematical program was introduced previously in the literature that can handle this problem in the way it is presented here. Most of the prior work in this area was done assuming that subassemblies have similar variations. In this work we handle the case of both similar and dissimilar subassembly variations. Almost all the work in the literature in this area has focused on simple assemblies, such as pin-hole assembly. Our model can handle more complicated assemblies.

The Chapter is divided into five sections and organized as follows: In Section 4.2, we introduce the problem statement. In Section 4.3, a mathematical model and the proposed heuristic are presented. Applicability for selective assembly is presented in the same section. Numerical examples are explained and solved in Section 4.4 for illustrative purposes. Finally, concluding remarks and findings are summarized in Section 4.5.

4.2. PROBLEM

Referring to Figure 4.1, suppose we are to produce two assemblies that are made of two subassemblies from groups* A and B (they have nominal dimension of 1 for each one of them). The assembly process is performed by placing a part from A and another from B in a bin that has an exact capacity of 2. If we arbitrarily chose to locate A_1 and B_1 in a bin and A_2 and B_2 in another bin, this will result in two "bad" final assemblies. However, if we assign A_1 with B_2, and A_2 with B_1, we will end up with two "perfect" final assemblies. The act of assigning these subassemblies together could be troublesome if we are dealing with a larger problem. The size of the problem is determined by the number of subassembly groups and the number of items in each group. In this Chapter, we propose a model that can be solved to optimally produce final assemblies with the least variation.

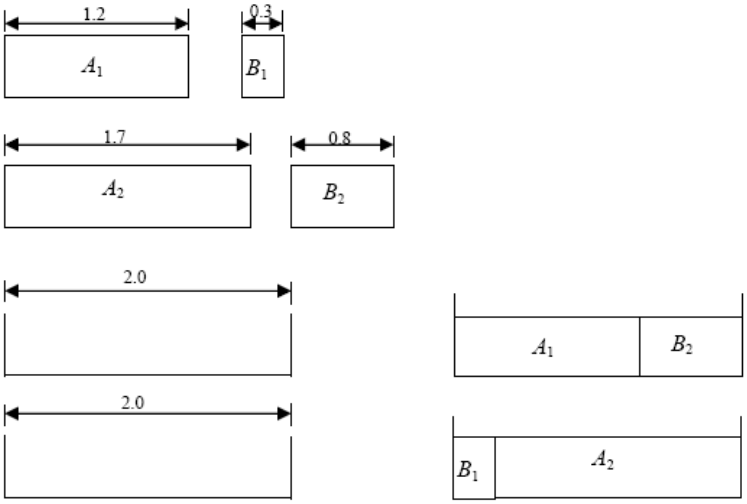


Figure 4-1. Assembly with two subassembly groups: A and B

4.3. APPROACH

4.3.1. Assumptions

* *Group* here refers to the set of parts in a subassembly. It does not refer to the partitioned group in selective assembly.

We assume the following in our model:

- All the subassembly measurements are taken for the whole batch, the batch size is P (number of final assemblies).
- The number of items in a group is equal to the batch size; $I = P$.
- There is no mean shift from the nominal dimensions of subassemblies. This assumption will be relaxed in the next Chapter (Chapter 5).
- The assembly functions (defined in Section 4.3.2) are linear.
- Our main objective is to minimize the variation no matter what are the final specifications for the final assemblies. This assumption will be relaxed in the next Chapter (Chapter 5) when we introduce the Dynamic Throughput Maximization (DTM) model.

4.3.2. Mathematical Model

Let us start defining what we mean by *assembly function* and *conflicting* and *non-conflicting assembly systems*.

Definition 1: Assembly Function
Assembly function is the mapping function between the subassembly quality characteristics (x's) and a final assembly quality characteristic (y's).

Definition 2: Assembly System
Assembly system is a collection of Q assembly functions: $y_1 = f(x_1,...,x_G), ..., y_Q = f(x_1,...,x_G)$

Definition 3: Conflicting vs. Non-Conflicting Assembly Systems
An assembly system is called *non-conflicting* if the variables in the right hand side (RHS) of *all* the equations have similar signs by either multiplying some equations by -1 and/or by keeping some of the equations as they are. On the other hand, an assembly system is called *conflicting* if the variables in the RHS of *all* the equations do not have similar signs even if some RHS's of the

equations are multiplied by -1. To understand why we multiply by -1, please refer to the following explanation.

Definition 2 can be further explained by the following examples: suppose we have two final quality characteristics (y_1 and y_2) and two subassembly groups (x_1 and x_2) that are mapped together as follows: $y_1 = x_1 + x_2$ and $y_2 = x_1 - x_2$. Notice that x_2 has a positive impact on y_1 and a negative impact on y_2, also notice that x_1 has positive impacts on y_1 and y_2. The Equations in this form show that we have different signs in the right hand sides of the Equations. We also cannot make the signs of the RHS of the Equations to be similar by multiplying the RHS of one Equation by -1. Therefore, we call this system *conflicting*. Two examples about the *non-conflicting* systems are: (1) $y_1 = 2x_1 + 7x_2$ and $y_2 = 10x_1 + 4x_2$ and (2) $y_1 = 3x_1 + 5x_2$ and $y_2 = -12x_1 - 20x_2$.

The mathematical model in this Chapter is valid for *additive dimensional* (or *functional*) assembly functions, where an additive chain function can be represented as follows:

$$y = x_1 + x_2 + \ldots + x_G \qquad (4.1)$$

Where,

y: quality characteristic of the final assembly

x_k: k^{th} Subassembly quality characteristic

It is worthy to mention that non-additive functions with different forms can be linearized by finding the logarithmic form of it. For example, a function that is given in Equation 4.2 can be linearized in the form shown in Equation 4.3). Other nonlinear forms can be dealt with differently as we will show later in this Chapter. One approach is to linearize the functions using Taylor series. This is expected to be accurate for our case as we are dealing with small ranges in tolerancing problems.

$$y = x_1 . x_2 / x_3 \qquad (4.2)$$

$$y' = x_1' + x_2' + x_3' \qquad (4.3)$$

Where:

$y' = \ln(y),$

$x' = \ln(x_1),$

$x_2' = \ln(x_2),$

$x_3' = -\ln(x_3)$

We aim to reduce the variation by approaching the nominal value of a quality characteristic q (nom_q) as much as possible for all the Q final quality characteristics of all the P final assemblies. In other words, the objective function and constraints can be formulated as follows:

$$Minimize \qquad F = \sum_{p=1}^{P}\sum_{q=1}^{Q}\left| nom_q - \sum_{g=1}^{G}\sum_{i=1}^{P} x_{gip} d_{giq} \right| \qquad (4.4)$$

Subject to: $$\sum_{i=1}^{P} x_{gip} = 1 \qquad \forall g = 1,...,G, \forall p = 1,...,P \qquad (4.5)$$

$$\sum_{p=1}^{P} x_{gip} = 1 \qquad \forall i = 1,...,P, \forall g = 1,...,G \qquad (4.6)$$

$$x_{gip} = \begin{cases} 1, \text{ if item } i \text{ in group } g \text{ is used to assemble part } p \\ 0, \text{ otherwise} \end{cases}$$

Constraints (4.5) imply that exactly one item (i) from a group (g) has to be selected to build an assembled part (p). Constraints (4.6) imply that exactly one assembled part (p) will be selected to include an item (i) in group (g).

Where:

g :Group index; $g \in \{1,...,G\}$

G :Number of subassembly groups

q : Quality characteristic index; $q \in \{1,...,Q\}$

p : Part index; $p \in \{1,...,P\}$

P : Batch size (Number of assembled parts)

d_{giq} : Dimension of item i in subassembly group g that is associated with quality characteristic q (Note: $d_{giq} = 0$ if group g does not contribute in quality characteristic q)

nom_q: Nominal value for quality characteristic q

Q: Number of final quality characteristic

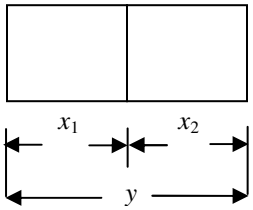
In order to linearize the objective function in (4.4), we introduce a new variable y_{pq}. This will modify the objective function as follows:

$$\text{Minimize} \qquad F = \sum_{p=1}^{P} \sum_{q=1}^{Q} y_{pq} \qquad (4.7)$$

Subject to the following additional sets of constraints:

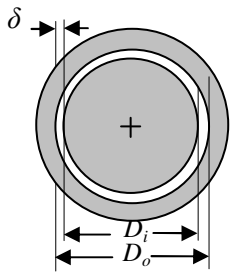
$$y_{pq} + \sum_{g=1}^{G} \sum_{i=1}^{P} x_{gip} d_{giq} \geq nom_q \qquad \forall p = 1,..., P, \forall q = 1,..., Q \qquad (4.8)$$

$$y_{pq} - \sum_{g=1}^{G} \sum_{i=1}^{P} x_{gip} d_{giq} \geq -nom_q \qquad \forall p = 1,..., P, \forall q = 1,..., Q \qquad (4.9)$$



$$y = x_1 + x_2$$

Figure 4-2. Additive assembly function



$$\delta = (D_o - D_i)/2$$

Figure 4-3. Subtractive assembly function

4.3.3. Heuristic

Special Case: Dynamic Variation Reduction for a Single quality characteristic (Q=1), and two subassembly groups (G=2):

Optimal solution for an assembly (C) made of two subassemblies (A and B; where the assembly function is $c=a+b$) for a single quality characteristic (Q=1) can be achieved by creating new groups of subassembly groups (A' and B') that are sorted in a complimentary manner, i.e. sort A from minimum to maximum and sort B from maximum to minimum. Then, we assign the first part in A' with the first part in B' and so on. Figure 4.4 shows a pictorial description of the aforementioned heuristic. This is consistent with what Pugh (1986) and Desmond and Setty (1962) mentioned about reducing the between-group variation by matching the small to large partitions. This is further explained in Remark 4.1. Figures 4.2 and 4.3 respectively show two examples when we have additive and subtractive assembly functions. In the case of the additive function, we actually assign the largest existing item in A with the smallest existing item in B until we finish assembling the P final assemblies. On the other hand, when we have the same rule for the subtractive assembly function, we will assembly the largest item in A (shaft) with the largest existing item in B (hole).

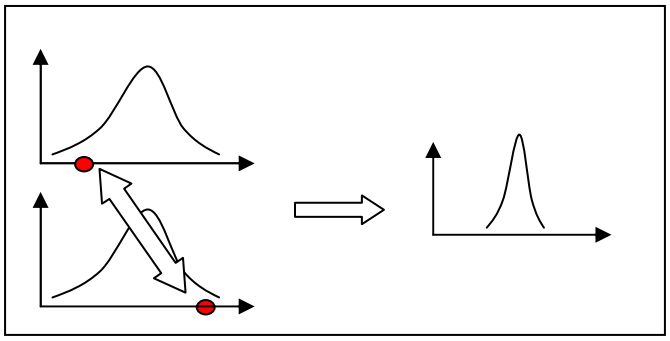


Figure 4-4. Variation reduction for additive function when Q=1 and G=2

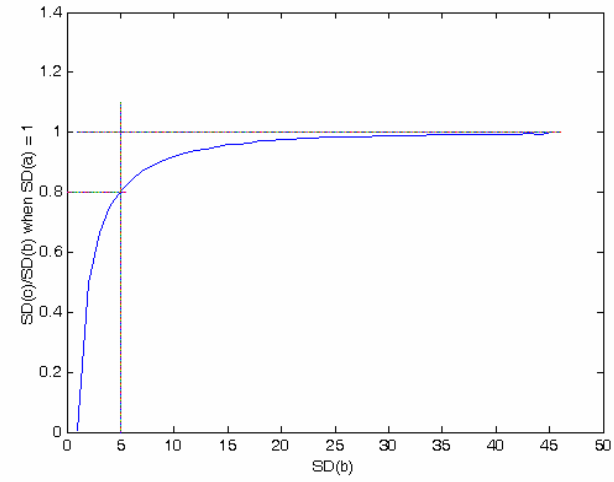


Figure 4-5. The effect of applying the rule when the variances of subassemblies are dissimilar; SD(c)/SD(b)=0.80 (Variation Reduction =20%) when SD(b)=5.SD(a); a and b follows symmetric PDF's with the same parameters

Figure 4.5 shows the effect of variance dissimilarity when assembling two parts (a and b) on the variation reduction of the final assembly (c) when considering our approach. The Figure suggests that the maximum variation reduction can be reached when the two subassemblies have the same variance. In theory, if the two subassemblies follow the same probability distribution, then the expected variation after applying the aforementioned rule approaches zero. Referring to Figure 4.5, we can see that if one of the subassemblies (b) has five times as much variation (standard deviation) in the other subassembly (a), then the expected variation reduction in c is only 20%. It is almost always correct that the standard deviation of a subassembly will be less than 5 times the standard deviation of the other subassemblies in the final assembly. You can also notice that the

variation reduction is expected to approach zero when the standard deviation of b is about 35 times more than the standard deviation of a. The reason behind that is that at that level of dissimilarity, the process of assigning a with b according to the rule is like adding a random variable (b) to a constant (a). The following remarks summarize our findings in regard of solving the proposed mathematical model.

Remark 4.1:

I. **(3-Rule Heuristic)** For a final assembly with one quality characteristics (Q=1) that is composed of G subassembly groups with *additive* assembly function:

(a) If G=2, the final assembly variation can be *significantly* reduced by sorting the P items ascendantly for the first subassembly group, descendantly for the second subassembly group, and then assigning the subassemblies respectively. The solution using the proposed approach for this case is optimal. This is proven in the Appendix.
(b) If G=3, the final assembly variation can be reduced by sorting the three subassembly groups and then following the rule explained in Figure 4.7.
(c) A final assembly with one quality characteristic (Q=1) with G>3 can be solved as a combination of rules (a) and (b) in I. Rules (a) and (b) have to be applied for subassembly groups with close variations because we know that we can make use of our dynamic variation reduction with the mated subassembly groups when their variations are close.

II. Increasing the number of subassembly groups (G) typically further reduces the variation when we use the aforementioned rules. This means that a product with $G+K$ subassembly groups with identical probability density functions usually has less variation that a product that is made of G subassembly groups (G>1 and $K\geq$1).

Remark 4.2

For a final assembly with multiple quality characteristics (Q>1), variation can be reduced by using the aforementioned rules if the subassembly quantities have the same proportionality effect on each other, i.e. assembly systems is non-conflicting.

Remark 4.3

If *A* and *B* follow identical symmetric probability density functions, applying rule I(a) will reduce the variance to 0 (See figure 4.6).

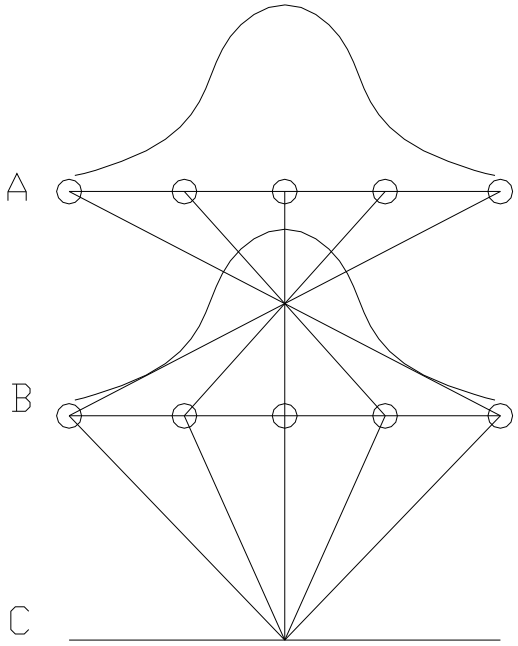


Figure 4-6. If *A* and *B* are identical and we have one assembly function, then applying the rules for many items will approach the variance to zero

Figure 4.7 explains rule I(b) for remark 4.1 when we have 3 subassembly groups for 7 items (G=3, P=7). After sorting the three subassembly groups from minimum to maximum; in step (1), we assign the first item in subassembly group 1 with the last (seventh) item in subassembly group 2 and the item in the middle (forth) in subassembly group 3. In step (2), we assign the first item in subassembly group 2 with the seventh item in subassembly group 3 and the forth item in

subassembly group 1. The same procedure is followed for step 3. Notice in step 4, we assign the second item in subassembly group 1 with the sixth item in subassembly group 2 and an item in the middle in subassembly group 3. This middle item is chosen to be the one that is *nearest* to the mean of the values in subassembly group 3. This same rule applies for steps 5 and 6. In step 7 where we create the seventh final assembly, we finish by choosing the remaining slots in the table as it is shown in Figure 4.7.

Example on Remark 4.2 is when two quality characteristics δ_1 and δ_2 (Q = 2) are the quality characteristics of the final assembly and the assembly functions are shown below (Figure 4.14). The three subassembly groups are: A, B and C. Notice that a, b and c are the subassembly quantities. Notice that a and b have negative impacts on δ_1, where c has a positive impact on δ_1. On the other hand, a and b have positive impact on δ_2, where c has a negative impact on δ_2. In summary, the assembly system is non-conflicting; therefore we can use the 3-Rule heuristic.

$$\delta_1 = f(a,b,c) = c - \sqrt{a^2 + b^2} \qquad \delta_2 = f(a,b,c) = a - \sqrt{c^2 - b^2}$$

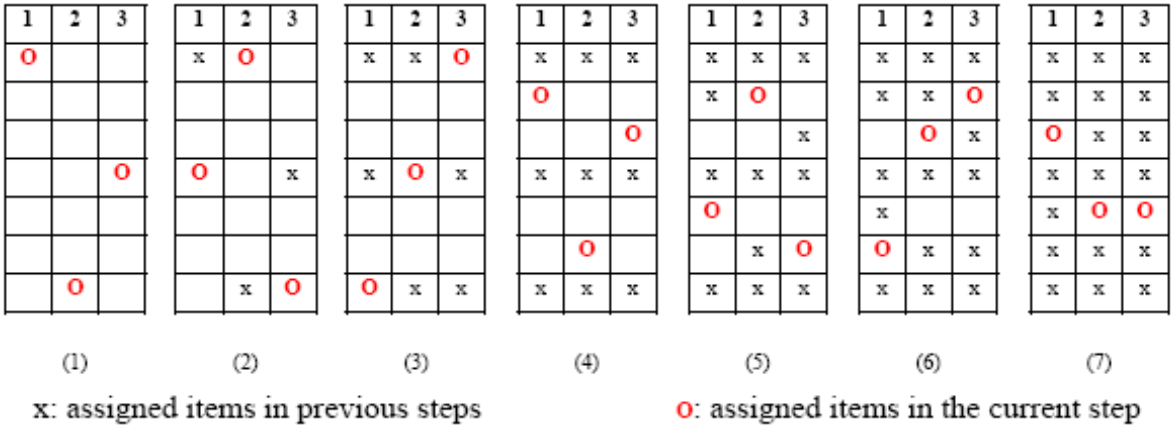


Figure 4-7. Rule I(b) in remark 4.1: allocate 7 items in 3 subassembly groups

Figure 4.8 shows an example of how we can solve the problem when we have G>3. In this example, we have 9 subassembly groups. There are two possibilities to solve that problem. In both cases, subassembly groups that are grouped together according to a rule (a or b) must have close variations to take the most advantage from the rules to reduce the variation. For instance;

the solution in the left hand side is best when: Var(g=1)≈Var(g=2), Var(g=3)≈Var(g=4), Var(g=5)≈Var(g=6), and Var(g=7)≈Var(g=8) ≈Var(g=9).

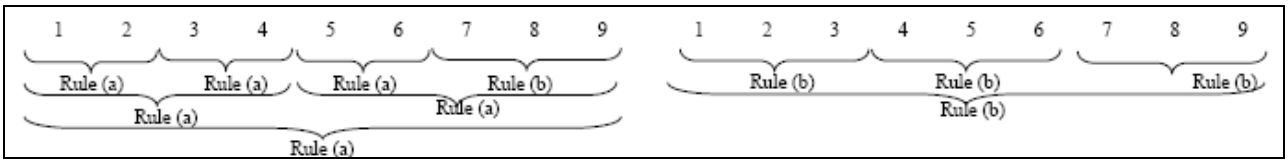


Figure 4-8. Heuristic implementation for 9 subassembly groups using a combination of rules (a) and (b)

Notice that for a subtractive function; such as: $c = HD - SD$ (c: Clearance; HD: Hole diameter; SD: Shaft diameter (Figure 4.3)), the rule still applies. However, in this case, the optimal arrangement will be to assign the largest shaft to the largest hole.

4.3.4. Applicability to Selective Assembly

The objectives of selective assembly are to find the optimal group partitions, number of subassembly groups and their matings. We can think of our model as a selective assembly problem with subassembly group size equal to the number of items in each subassembly group. The available approaches in the literature deal with the problem after assuming that the subassembly item dimensions are normally distributed. Also, it is commonly assumed to deal with subassembly groups with similar variation. We propose a model that is based on simulation that does not need to assume any kind of distribution or distribution parameters to solve the problem.

A flowchart that depicts the proposed approach is presented in Figure 4.9. Initially, we generate I random variables for each subassembly based on their probability distribution functions and their parameters. The probability distribution functions can be normal, uniform, etc. Then, we solve for the best matings of those virtual parts by using aforementioned rules introduced in Section 4.3. We know at this point that we reach the least assembly variation when we develop a plan based on the part level. Notice that when the number of partitioned groups is I (number of items in each subassembly group), we achieve the least possible variation (refer to Figure 4.10). In

contrast, having one group (single bin) of all subassembly parts means that the assembly is arbitrary over the whole population which will lead to the highest possible variation. This is represented by Taguchi Loss cost function that is represented in Equation 4.10.

$$C_{Variation} = M.\sigma^2 \qquad (4.10)$$

Where:

M: Cost of variation; [\$/unit^2]

σ^2: Variance of the concluding link in an assembly; [unit]

As we mentioned earlier in Section 4.3.1, we assume that there is no drift from the nominal value, therefore the only term of Taguchi Loss function we will have here is the variance. The partitioning cost, that is also pictorially shown in Figure 4.10 and expressed in Equation 4.11, balances the effect of the Taguchi Loss function.

$$C_{\text{Partitioning}} = CP.(NP-1) \qquad (4.11)$$

Where:

NP: Number of partitioned groups; [groups]. $NP \in \{1,...,I\}$

I: Maximum number of groups [groups]

CP: Cost associated with adding a single group [\$/group]

In our approach, we start with a number of partitions that is equal to the number of parts in a subassembly (I). We improve the solution by reducing the total cost when reducing the number of partitions. We stop if the cost of the current plan is larger than the cost of the previous plan or if the number of partitions is zero (one group). Achieving a solution with one group indicate that it is better to interchangeably and non-selectively assemble the parts.

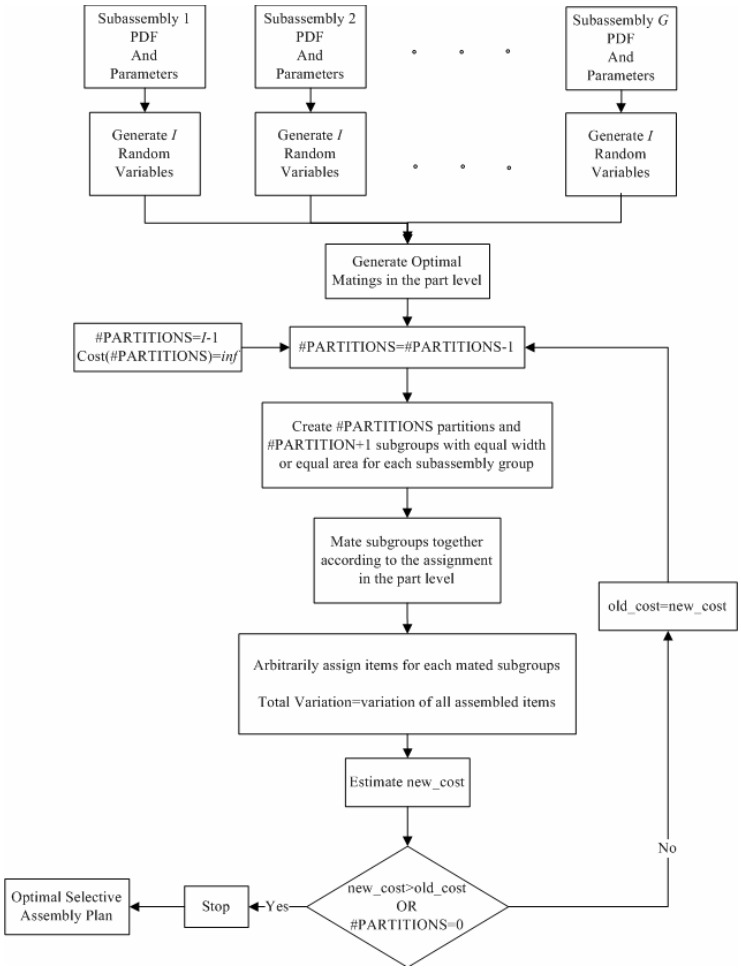


Figure 4-9. Selective assembly approach

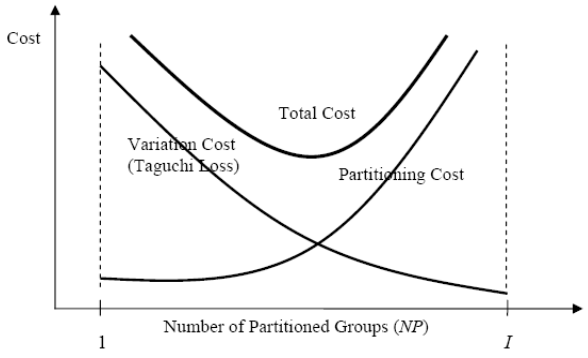


Figure 4-10. Cost behavior vs. number of partitioned groups (G) in selective assembly

Figure 4.11 depicts the cost behavior when changing the number of partitioned groups (G) in selective assembly for a final assembly with two subassemblies (I=1000) with identical and

symmetric *PDF*'s and additive assembly function. Notice that selective assembly quality characteristics would be within the bounds of arbitrary assembly and dynamic assembly.

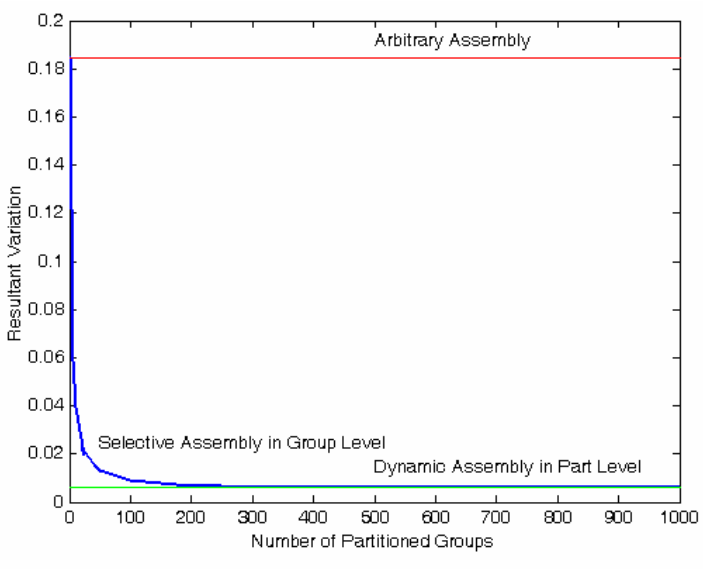


Figure 4-11. Effect of increasing partitions on reducing variation for $c=f(a,b)$, Arbitrary Assembly refers to a bin size of 1, Selective Assembly refers to 6 number of bins, ad dynamic assembly refers to *I* number of bins

4.4. NUMERICAL EXAMPLES

Example 1

This example was taken from Kannan et al. (2003) paper. They used a Genetic Algorithm to solve the problem of equal-width selective assembly. It is a problem with three blocks with dimensions of 10+0.012 to be placed next to each other so the variation of their length is to be minimized. They considered 6 bins, and the range of final dimension was decreased from 36 μ to 8 μ. Their approach is based on worst case variation estimation. We used our rule (b) for solving the same problem for equal-area selective assembly for 6 bins. Notice we did not follow the procedure in the flowchart (Figure 4.9) because we are solving to compare the results with different work that used 6 number groups. Based on a simple implementation of our heuristic, we

got a smaller range of final dimension (7.523 μ). The bins are depicted in Figure 4.9 and the selective assembly plan based on our heuristic is shown in Table 4.1.

Table 4.1. DVR solution

Group Number	***a***	***b***	***c***
1	1	6	3
2	4	1	4
3	6	3	1
4	2	5	4
5	3	2	5
6	5	4	2

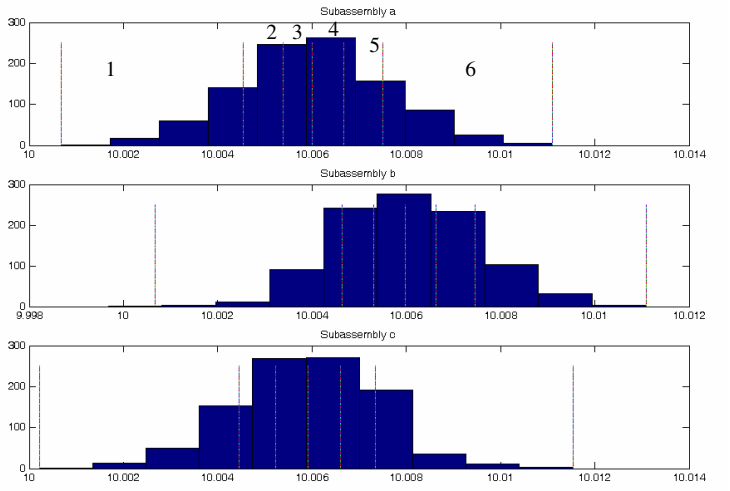


Figure 4-12. Subassembly distributions and groups for selective assembly with 6 bins for Example 1

We also found that using our approach will give a ratio of SD(*arbit*)/SD(*SA*)=2.28 and SD(*arbit*)/SD(*DVRT*)= 6.729, where *arbit* refers to the arbitrary assembly, *SA* refers to the selective assembly with six bins and *DVRT* refers to Dynamic Variation Reduction Technique that is based on the part level. Figure 4.13 compares between the three approaches. Notice that after virtually assembling product number 100 (Figure 4.13), the shape of the curve of the DVRT

assembly takes a harmonic form with increasing amplitude. Before that, we notice that the behavior is different. This implies that by increasing the number of mated items according to our rule, we expect to have higher variation. This finding contradicts with Remark 4.1(II), which suggests that our rule for G=3 (3 subassembly groups) is not necessarily an optimal solution. This interesting harmonic behavior can be observed when we apply our rules for 3 subassembly groups (rule 1(b)) that is explained in Figure 4.7. The reason behind that is that at the beginning, we mate items that are located on the tails (most probably outliers) of the three distributions and later we assign items that are located near the mean. The change in the amplitude is case-specific.

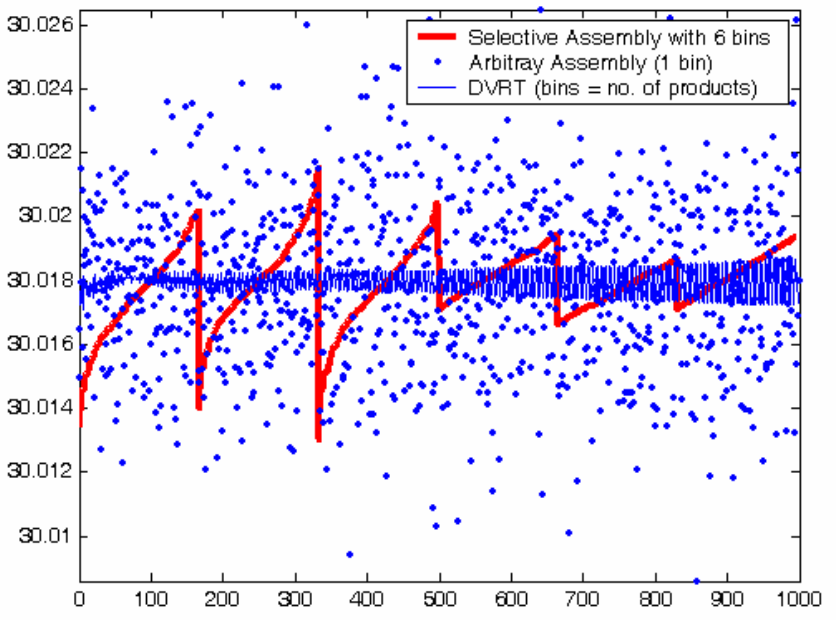


Figure 4-13. Variation of final assembly using *selective*, *arbitrary* and *DVRT* assembly for Example 1

Example 2

We now present an example for a 3-bar assembly with 2 quality characteristics (Figure 4.14) where the objective function is nonlinear. In the problem; a, b and c are the subassembly quality characteristics and δ_1 and δ_1 are the final quality characteristics to be satisfied (Equations 4.12). The objective function is shown in Equation 4.13. Therefore, we propose solving the problem by linearizing the objective function by using Taylor series then by using our 3-Rule heuristic. We solved the problem using a GA and we used it as a benchmark to compare it with our heuristic.

$$\begin{aligned}\delta_1 &= f(a,b,c) = c - \sqrt{a^2 + b^2} \\ \delta_2 &= f(a,b,c) = a - \sqrt{c^2 - b^2}\end{aligned} \qquad (4.12)$$

We assume that: $a,b \sim N\left(2,0.10^2\right)$ and $c \sim N\left(2.8,0.10^2\right)$. The nominal values for δ_1 and δ_1 are zeros.

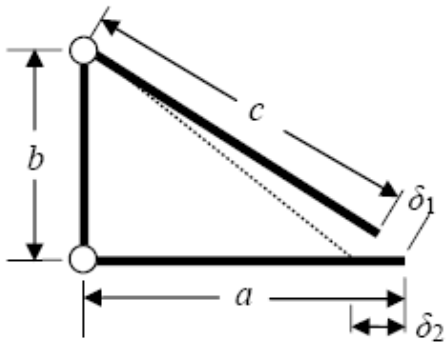


Figure 4-14. Assembly example with 2 quality characteristics (δ_1 and δ_2), 3 subassembly groups (a, b and c), nonlinear relationship

$$\text{Minimize } F = \sum_{p=1}^{P}\left(\left|\sum_{i=1}^{P} x_{cip}c_i - \sqrt{\sum_{i=1}^{P} x_{aip}{a_i}^2 + \sum_{i=1}^{P} x_{bip}{b_i}^2}\right| + \left|\sum_{i=1}^{I} x_{aip}a_i - \sqrt{\sum_{i=1}^{P} x_{cip}{c_i}^2 - \sum_{i=1}^{P} x_{bip}{b_i}^2}\right|\right) \quad (4.13)$$

$$x_{gip} = \begin{cases}1, \text{if item } i \text{ in group } g \text{ is used to part } p \\ 0, \text{otherwise}\end{cases} \qquad \forall g = a,b,c$$

Subject to: $\sum_{i=1}^{P} x_{gip} = 1 \quad \forall g = a,b,c; \forall p = 1,...,P$

$$\sum_{p=1}^{P} x_{gip} = 1 \qquad \forall i = 1,...,P; \forall g = a,b,c$$

Use Taylor series to linearize Equations in 4.12:

$$\delta = f(\bar{a},\bar{b},\bar{c}) + (a-\bar{a})\frac{\partial\delta(\bar{a},b,c)}{\partial a} + (b-\bar{b})\frac{\partial\delta(a,\bar{b},c)}{\partial b} + (c-\bar{c})\frac{\partial\delta(a,b,\bar{c})}{\partial c} + H.O.T \qquad (4.14)$$

$$\delta_1 \cong \underbrace{\left(\frac{-\bar{a}}{\sqrt{\bar{a}^2+\bar{b}^2}}\right)}_{\alpha_1} a + \underbrace{\left(\frac{-\bar{b}}{\sqrt{\bar{a}^2+\bar{b}^2}}\right)}_{\beta_1} b + \underbrace{c}_{\gamma_1} \rightarrow \delta_{1i} \cong \underbrace{\left(\frac{-\bar{a}}{\sqrt{\bar{a}^2+\bar{b}^2}}\right)}_{\alpha_{1i}} a_i + \underbrace{\left(\frac{-\bar{b}}{\sqrt{\bar{a}^2+\bar{b}^2}}\right)}_{\beta_{1i}} b_i + \underbrace{c_i}_{\gamma_{1i}} \qquad (4.15)$$

$$\delta_2 \cong \underbrace{a}_{\alpha_2} + \underbrace{\left(\frac{\bar{b}}{\sqrt{\bar{c}^2-\bar{b}^2}}\right)}_{\beta_2} b + \underbrace{\left(\frac{-\bar{c}}{\sqrt{\bar{c}^2-\bar{b}^2}}\right)}_{\gamma_2} c \rightarrow \delta_{2i} \cong \underbrace{a_i}_{\alpha_{2i}} + \underbrace{\left(\frac{\bar{b}}{\sqrt{\bar{c}^2-\bar{b}^2}}\right)}_{\beta_{2i}} b_i + \underbrace{\left(\frac{-\bar{c}}{\sqrt{\bar{c}^2-\bar{b}^2}}\right)}_{\gamma_{2i}} c_i \qquad (4.16)$$

Hence, the objective function is linearized as follows:

$$\text{Minimize } F = \sum_{p=1}^{P}\left(\left|\sum_{i=1}^{P}\left(x_{\alpha ip}\alpha_{2i} + x_{\beta ip}\beta_{1i} + x_{\gamma ip}\gamma_{1i}\right)\right| + \left|\sum_{i=1}^{P}\left(x_{\alpha ip}\alpha_{2i} + x_{\beta ip}\beta_{2i} + x_{\gamma ip}\gamma_{2i}\right)\right|\right)$$

$$x_{gip} = \begin{cases} 1, \text{if item } i \text{ in group } g \text{ is used to part } p \\ 0, \text{otherwise} \end{cases} \qquad \forall g = \alpha, \beta, \gamma$$

Subject to: $\sum_{i=1}^{P} x_{gip} = 1 \qquad \forall g = \alpha, \beta, \gamma; \forall p = 1,..., P$

$$\sum_{p=1}^{P} x_{gip} = 1 \qquad \forall i = 1,..., P; \forall g = a, b, c$$

Equations 4.15 and 4.16 indicate that we have a non-contradicting assembly system (defined in Section 4.3.2); therefore we can apply our proposed rules. Figures 4.15 and 4.16 show the solution for the two quality characteristics using our proposed heuristic. The found objective function value using our heuristic was found to be 8.56. When the problem was solved using a GA, the minimum objective function was found to be 6.56. We ran GA for 200,000 replications; using a 6-chromosome population and 10% inverse mutation rate (solution is shown in Figure 4.17). The execution time for a code in MATLAB was 116 minutes on 2.4 GHz Pentium 4, 512 MB processor. Inverse mutation is defined in Gen and Cheng (1997) as the act of selecting two genes in a chromosome and then swapping their values.

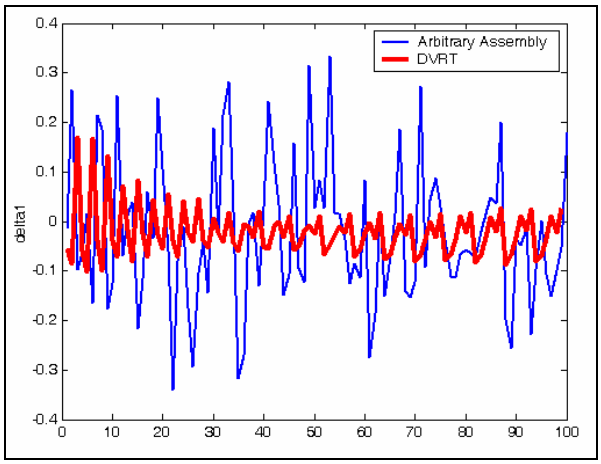


Figure 4-15. Arbitrary assembly vs. DVRT assembly considering δ_1 only

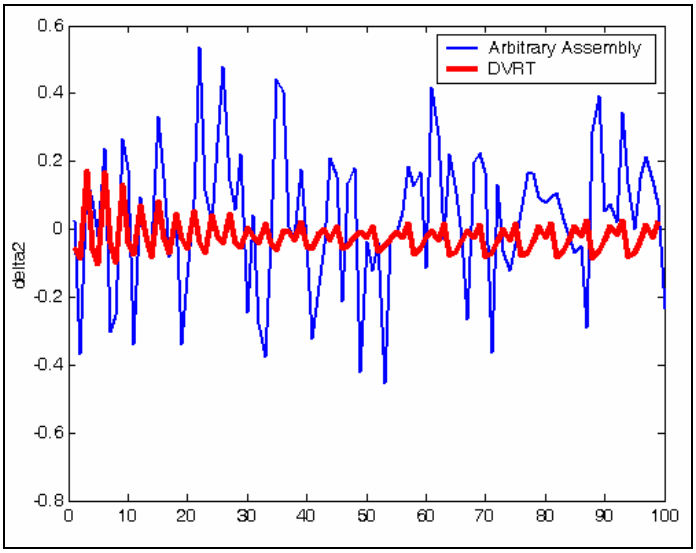


Figure 4-16. Arbitrary assembly vs. DVRT assembly considering δ_2 only

4.5. SUMMARY

We can summarize our findings and concluding remarks as follows:

(1) The inspection data for the subassemblies can be utilized to dynamically reduce variation for batches by optimally assigning subassemblies that will lead to the minimum possible variation (dispersion around the nominal values of the final assembly quality characteristics).

We proposed a mathematical model and different approaches to solve it to reduce the variation of the final assemblies for different assembly functions.

(2) For an additive assembly function ($c = a + b$), we found that the variation can be reduced optimally when we assign a's and b's according to the proposed rule. We assign the largest existing subassembly item in group A to the smallest existing subassembly item in group B.

(3) We found that the variation can be maximally reduced when the subassembly variations are close.

(4) For an additive assembly function; if we use the proposed rule for two identical probability density functions, we will theoretically produce a final assembly with *no* variation (if the variations for a and b are the same and have the same probability density functions (*PDF*)).

(5) The rule was proven to be optimal for an additive assembly functions with two groups and one quality characteristics. Also, it was verified (Example 2 in this Chapter) to produce near-optimal solution for more complicated assembly functions (3 groups with nonlinear assembly functions and two quality characteristics) by comparing it to a GA solution.

(6) We proposed a more general approach for selective assembly based on the dynamic variation reduction simulation.

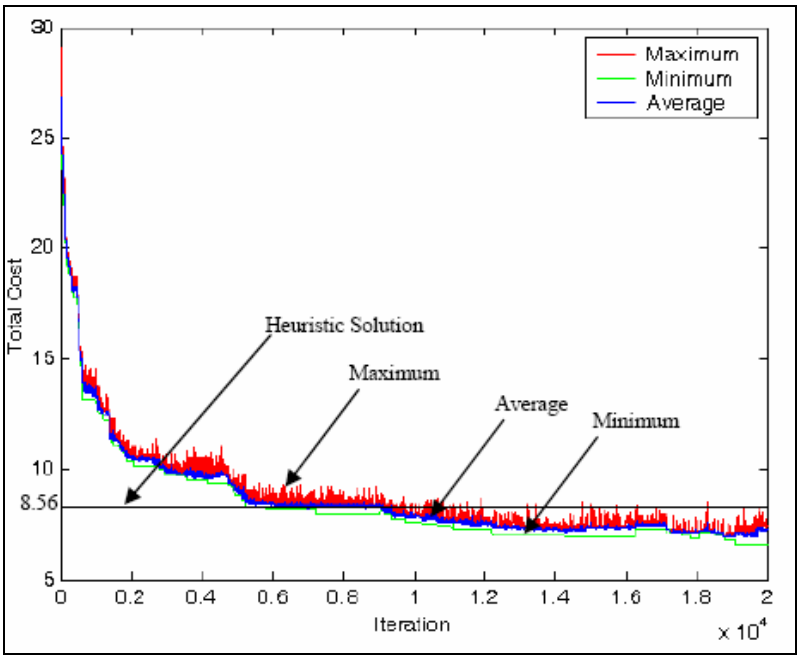


Figure 4-17. GA solution (Minimum, Maximum and Average solutions for all the Generations) using a typical cross-over and inverse mutation, mutation rate = 10%

(6) In our model, we considered minimizing the drift from the mean (Equation 4.4) over a batch of parts. A possible extension is to add another term associated with the spread around the mean. We present that in Chapter 5. This will make the objective function more analogous to Taguchi Loss Function. Unlike Taguchi Loss Function, we consider the absolute difference between the final quality characteristics and the nominal and mean values.

CHAPTER 5: ENHANCED DYNAMIC VARIATION REDUCTION (EDVR) AND DYNAMIC THROUGHPUT MAXIMIZATION (DTM) MODELS

5.1. INTRODUCTION

The need for variation reduction and rolled-yield-throughput maximization arise in various scenarios. We propose a mathematical model to match parts together in order to achieve the minimum possible variation of producing a batch in an assembly line. In this model, we minimize the drift from the nominal value and the variation around the mean (extension to the model presented in Chapter 4). The sorting rule for two subassembly groups presented in Chapter 4 is proven to be optimal for a special case. This is also true after adding the second term to the objective function; which accounts for the spread around the mean. Additionally, we propose a model that can solve the problem with the objective of maximizing the rolled-yield - throughput. Numerical examples are solved for illustrative and demonstrative purposes. Due to the increasing competition in manufacturing, solving those models have to be fast and efficient. Therefore, we propose to solve them in real-time basis.

In this Chapter, we address two questions. First, we investigate the best subassembly mating (assignment) that can reach the least variation (spread around the mean and drift from the nominal value). This is an extension of the work proposed in Chapter 4. In Chapter 4; we aimed to reduce the drift from the nominal value, whereas the present model incorporates a reduction of the spread around the local mean as well. This is more analogous with *Taguchi Loss Function*. Besides dynamic variation reduction, this model is also relevant when we do *not* have particular specifications for quality characteristics. Rather, we build this model to know how good we can do in terms of minimizing the variation from the mean and from the nominal value. This justifies for the assumed case (as in Section 5.3) where we do not have a tolerance that we need to control in our variation reduction optimization model. Second, we seek the best subassembly mating (assignment) that can achieve the maximum rolled-yield-throughput. A possible scenario where this question might be needed to be answered is when we are asked by a customer to satisfy

specifications for quality characteristics. We provide mathematical models for both cases and solve the first problem optimally using the simple heuristic (3-Rule Heuristic) presented in Chapter 4 for a special case. In the rest of this Chapter, we refer to the Rolled-Yield-Throughput by throughput for simplicity.

The rest of this Chapter is organized as follows: in Section 5.2, we explain the problem and its significance. In Section 5.3, an enhanced mathematical model for a previous model in Chapter 4 is proposed. We also present a mathematical model to maximize the throughput given that the final quality specifications are given by the customer (designer). Numerical examples are illustrated and solved in Section 5.4 for clarifying and illustrative purposes. Finally, concluding remarks and findings are summarized in Section 5.5.

5.2. PROBLEM

Suppose we have a simple assembly that is made of two subassemblies: A and B. A and B have a nominal dimension of 1. The assembly process is performed by placing one part from A and another from B in a bin that has an exact capacity of 2 (Figure 4.1). If we arbitrarily chose to locate A_1 and B_1 in a bin and A_2 and B_2 in another bin, this will result in two "bad" final assemblies because the final dimensions for the assemblies will be 1.5 and 2.5 which are far off from 2. The first one will occupy only 75% of the bin and the second one will over-occupy the bin by 25%. Alternatively, if we assign A_1 with B_2, and A_2 with B_1, we will end up with two "perfect" final assemblies. The act of assigning these subassemblies together is computationally challenging if we are dealing with a larger problem. In this Chapter, we propose two mathematical models that can be solved to optimally produce final assemblies with the least variation and drift from the mean and another one that can produce the maximum throughput when the quality characteristic specifications are given.

5.3. APPROACH

We assume the following in our model:

- All the subassembly measurements are taken for the entire batch, and the batch size is P.
- The number of contributing items in a subassembly group (I) is equal to the batch size; i.e. $I = P$.
- The following assumption is valid for our Dynamic Variation Reduction Model and not valid for the Dynamic Throughput Maximization model. There is no constraint on the final assembly dimensions for the variation reduction problem. See the set of constraints in Equations 5.5 that represents that. This is analogous with the selective assembly assumptions.
- Quality characteristics are not correlated.
- Any assembly function can be modeled as an additive function. The mathematical model in this Chapter is valid for additive dimensional (or functional) chains, where an additive chain function can be represented as follows:

$$CL = CO_1 + CO_2 + ... + CO_n \qquad (5.1)$$

Where,
CL: Concluding link in the chain
CO_k: k^{th} Contributing link in the chain

5.3.1. Enhanced Dynamic Variation Reduction (EDVR) Model

We aim to reduce the variation by approaching the nominal value of a quality characteristic (nom_q) and by *thinning* the spread around the mean as much as possible. In other words, the objective function can be formulated as shown in Equations (5.2 to 5.4). The first term in the objective function accounts for the drift from the nominal value (presented in Chapter 4 in Equation 4.4) and the second term accounts for the spread around the local mean. In some sense, this function is comparable to *Taguchi Loss Function*. Here, we assume that both terms have the same importance.

$$\text{Minimize } F = \sum_{p=1}^{P}\sum_{q=1}^{Q}\left(\left|nom_q - \sum_{g=1}^{G}\sum_{i=1}^{P} x_{gip} d_{giq}\right| + \left|\sum_{g=1}^{G}\sum_{i=1}^{P} x_{gip} d_{giq} - (1/P)\sum_{g=1}^{G}\sum_{i=1}^{P} d_{giq}\right|\right) \quad (5.2)$$

Subject to:
$$\sum_{i=1}^{P} x_{gip} = 1 \qquad \forall g = 1,...,G\,, \forall p = 1,...,P \quad (5.3)$$

$$\sum_{p=1}^{P} x_{gip} = 1 \qquad \forall i = 1,...,\ P\ , \forall g = 1,...,G \quad (5.4)$$

Where:

$$x_{gip} = \begin{cases} 1, \text{if item } i \text{ in group } g \text{ is used for part } p \\ 0, \text{otherwise} \end{cases}$$

i: Item index; $i \in \{1,...,P\}$

g: Group index; $g \in \{1,...,G\}$

q: Quality characteristic index; $q \in \{1,...,Q\}$

p: Part index; $p \in \{1,...,P\}$

d_{giq} : Dimension of item i in group g that is associated with quality chrematistic q

Note: $d_{giq} = 0$ if group g does not contribute in quality characteristic q

nom_q: Nominal value for quality characteristic q

Constraints (5.3) and (5.4) are typical assignment constraints. Constraints (5.3) imply that exactly one item (i) from a group (g) has to be selected to build a part (p). Constraints (5.4) guarantee that an item (i) from any group (g) should be included in one and only one part (p).

3-Rule Heuristic

This heuristic is explained in details in Chapter 4. It was found that even with the addition of the second term in the objective function; rule I(a) is still optimal. This is proven in the appendix.

5.3.2. Dynamic Throughput Maximization (DTM) Model

The following mathematical program aims to maximize the rolled-yield-throughput (for short: throughput) of final assemblies under tolerance constraints (Equation 5.9).

$$\text{Maximize} \quad P' = \sum_{i=1}^{P} \sum_{p=1}^{P} x_{1ip} \tag{5.5}$$

Subject to:

$$\sum_{i=1}^{P} x_{gip} \le 1 \qquad \forall g = 1,...,G\,, \forall p = 1,...,P \tag{5.6}$$

$$\sum_{p=1}^{P} x_{gip} \le 1 \qquad \forall g = 1,...,G\,, \forall p = 1,...,P \tag{5.7}$$

$$\sum_{i=1}^{P} x_{gip} = \sum_{i=1}^{P} x_{g+1,ip} \qquad \forall g = 1,...,G-1\,, \forall p = 1,...,P \tag{5.8}$$

$$LL_q \sum_{i=1}^{P} x_{1ip} \le \sum_{g=1}^{G} \sum_{i=1}^{P} x_{gip} d_{giq} \le UL_q \sum_{i=1}^{P} x_{1ip} \qquad \forall q = 1,...,Q\,, \forall p = 1,...,P \tag{5.9}$$

The mathematical model is shown in Equations (5.5-5.9). The decision variables in the model are $x_{gip}\ (\forall g = 1,...,G, \forall i = 1,...,P, \forall p = 1,...,P)$. They are all binary variables (0's or 1's). A decision variable is 1 if item *i* in subassembly group *g* was decided to be used to make part *p*; and it is 0 otherwise. The objective (Equation 5.5) of this model is to maximize the number of "good" produced final assemblies (P') out of the total number of subassemblies (*P*) in each group. This is represented by the total number of items from the first group (*g*=1) that lead to correct final assemblies ($\sum_{i=1}^{P} \sum_{p=1}^{P} x_{1ip}$). The first group was arbitrarily chosen because we know that: $P' = \sum_{i=1}^{P} \sum_{p=1}^{P} x_{1ip} = \sum_{i=1}^{P} \sum_{p=1}^{P} x_{2ip} = ... = \sum_{i=1}^{P} \sum_{p=1}^{P} x_{Gip}$. Notice that $P' \le P$, so the throughput can be defined as P'/P. We dropped the denominator (*P*) because it is constant. Constraints (5.6) indicate that at most one item (*i*) in a group (*g*) is chosen for a product (*p*). Similarly, Constraints (5.7) indicate

that any item (i) from group (g) can be included in at most one part (p). Constraints (5.8) state that if a product was decided to be made, it should be fully assembled and must contain one subassembly item from each group. This constraint prevents the construction of partial or incomplete parts. Finally, Constraints (5.9) indicate that the dimensional characteristic of quality characteristic q for each *chosen* product p must be bounded between lower and upper limits (LL_q, UL_q) given that we have linear assembly systems. Notice that $\sum_{g=1}^{G}\sum_{p=1}^{P} x_{gip}.d_{giq}$ is an additive (linear) function. This formulation is different from the tolerance allocation problem. The reason is that the objective from the tolerance allocation is to specify limits in the subassemblies for the acceptance range that will possibly lead to the least failed final assemblies. Final assemblies fail if the final part is not within the tolerances specified by the customer (designer).

5.4. NUMERICAL EXAMPLE

In this example, we examine the differences in the results if we assembled a batch of 10 final assemblies (made of two subassembly groups: a and b) arbitrarily and based on the DTM model. The tolerance specifications for the quality characteristic (C) are: UL=20.2 and LL=19.8. The results are depicted in Tables 2 and 3. We solved the model using CPLEX solver that uses Branch and Bound algorithm. The DTM assembly increases the throughput 7 times compared to the solution provided by the arbitrary assembly. Moreover, the variation decreases 13.7 times.

5.5. SUMMARY

The inspection data for the subassemblies can be utilized to dynamically reduce variation or to increase rolled yield throughput for batches by optimally assigning (mating) subassembly items. We proposed two mathematical models (Enhanced Dynamic Variation Reduction to the one presented in Chapter 4 and Dynamic Throughput Maximization). We proposed simple rules to solve the Dynamic Variation Reduction (DVR) models. Those rules are also valid for the extended model we presented in this Chapter. However, for the Dynamic Rolled Yield

Throughput (DTM) model, we presented the mathematical program (Integer Program) without any heuristics or rules. Solving the IP is usually time-consuming for a problem instances with practical size. In order to achieve fast solutions, we are presenting different heuristics (Simulated Annealing, Ant Colony and Greedy Heuristics) in Chapter 6.

Table 5.1. Arbitrary Assembly

a	***b***	***c***	***c* within Tolerance?**
10.6353	9.604	20.2393	No
9.3986	10.6352	20.0338	Yes
10.5512	10.361	20.9122	No
8.9002	9.3366	18.2368	No
10.086	11.6837	21.7697	No
7.9954	10.4846	18.48	No
9.5069	11.0362	20.5431	No
10.462	9.7836	20.2456	No
9.679	9.8444	19.5234	No
11.2366	10.8144	22.051	No

Table 5.2. DTM Assembly

a	***b***	***c***	***c* within Tolerance?**
8.9002	11.0362	19.9364	Yes
10.6353	9.3366	19.9719	Yes
9.3986	10.4846	19.8832	Yes
9.5069	10.6352	20.1421	Yes
10.086	9.8444	19.9304	Yes
10.462	9.604	20.066	Yes
9.679	10.361	20.04	Yes
-	-	-	-
-	-	-	-
-	-	-	-

CHAPTER 6: METAHEURISTICS ALGORITHMS FOR THE DYNAMIC THROUGHPUT MAXIMIZATION PROBLEM

6.1. INTRODUCTION

Given that we have inspected a batch of subassembly items, it is sometimes more cost-effective to exploit the data of the measured features of the subassemblies to further reduce the variation in the final assemblies so rolled-yield throughput is maximized. This can be achieved by selectively assembling the subassembly items so we can maximize the rolled-yield-throughput of the final assemblies in a dynamic-basis. We introduced the Dynamic Throughput Maximization problem (DTM) in Chapter 5. The problem is found to grow substantially by increasing the size of the assembly (number of subassembly items and number of subassembly groups). Therefore, we resort to five algorithms in this Chapter: simple greedy sorting algorithm, two Simulated Annealing (SA) algorithms and two Ant Colony Optimization (ACO) algorithms. We solve numerical examples to compare the performances of the proposed algorithms.

The rest of the Chapter is organized as follows: in Section 6.2, we explain the problem and the mathematical model of the Dynamic Throughput Maximization and its solvability. Next, we present the five heuristics to solve the problem in Section 6.3. The heuristics are: greedy sorting rule, two Simulated Annealing Algorithms and two Ant Colony Optimization algorithms. In Section 6.4, we solve numerical examples to compare the proposed algorithms in terms of their computational time and the achieved objective function (throughput). In Section 6.5, concluding remarks and summary for our work are presented. Finally, some future research ideas related to this research are outlined in Section 6.6.

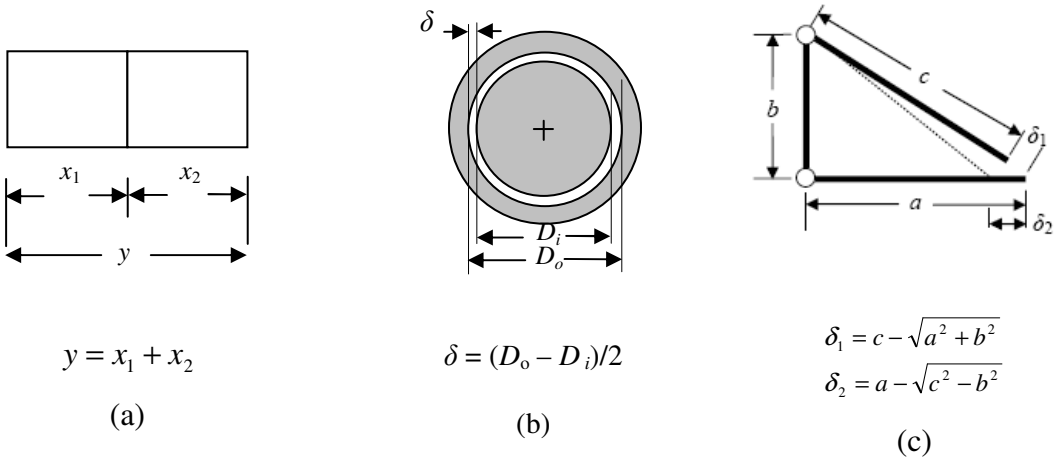
6.2. DYNAMIC THROUGHPUT MAXIMIZATION PROBLEM

6.2.1. Problem Statement

Assume a manufacturer is to assemble a final product so the *Rolled Yield Throughput* is to be maximized. Rolled Yield Throughput refers to the percentage of final assemblies that meet the Q

final assembly requirements (specifications). Each final assembly comprises G number of subassembly groups, and each subassembly group contains P number of items. The mapping assembly functions between the final assembly quality characteristics (y_q's) and the subassembly quality characteristics (x_g's) can take different forms. It can be linear or non-linear. Examples of linear and nonlinear subassembly functions are shown in Figures 6.1(a), 6.1(b) and 6.1(c).

The objective of this Chapter is to solve the *Dynamic Rolled Yield Throughput Maximization* (DTM) Problem. DTM refers to the act of mating subassembly together in near-real-time basis so we maximize throughput after having subassemblies inspected. The size of the problem grows substantially by increasing the number of subassembly items (P) and the number of subassembly groups (G). If we have decided to explore all the possible enumerations, then we will have to run $(P!)^{G-1}$ possible solutions. For example, a simple problem with 4 subassembly groups and 10 items in each group will require running $4.8\text{x}10^{19}$ enumerations. In the following section, we provide the mathematical problem for the problem as an Integer Program (IP) when the assembly function is linear (Figures 6.1(a) and 6.1(b)).



$y = x_1 + x_2$

(a)

$\delta = (D_o - D_i)/2$

(b)

$\delta_1 = c - \sqrt{a^2 + b^2}$

$\delta_2 = a - \sqrt{c^2 - b^2}$

(c)

Figure 6-1. (a): Two bock assembly; linear, (b): Pin-hole assembly; linear, (c): 3-Bar triangular assembly; nonlinear

6.2.2. DTM Mathematical Model

The mathematical model is presented in Chapter 5.

6.3. APPROACH

The problem is NP-hard (all Integer Programs (IP's) are NP-hard problems [Tovey (2002)]), therefore solving it optimally is challenging and time-consuming. We have resorted to heuristics because our aim is to find the quickest solution possible. We present five algorithms to solve the problem. The first algorithm in Section 6.3.1 is a greedy that is based on modifying simple rules we presented in an earlier publication for Dynamic Variation Reduction (DVR) in Chapter 4. The rest are metaheuristics: two Simulated Annealing algorithms in Section 6.3.2 and two Ant Colony Optimization algorithms in Section 6.3.3. Depending on the number subassembly groups and the nature of the system of assembly functions (linear vs. nonlinear), the proposed algorithms have different capabilities for solving DTM problem instances.

6.3.1. Greedy Sorting Algorithm

This algorithm is valid for non-conflicting, linear assembly systems with two subassembly groups (G=2). It is based on a previously proposed algorithm to solve the Dynamic Variation Reduction (DVR) problem in Chapter 4. For two subassembly groups (A and B), the spreads around the mean and the nominal values are minimized (proven to be optimal by contradiction when there is one final quality characteristic; Q=1 and G=2) if we assign the largest existing item in A to the smallest existing item in B until all the items are assembled. There are other rules proposed in Chapter 4 when we have 3 subassembly groups (G = 3) and when we have subassembly groups with different variations. Applying those rules when we seek maximizing the throughput does not guarantee "good" solutions because of possible shifts from the nominal values in the subassembly groups. Therefore, we revise the rules greedily for two subassembly groups (A and B). The idea is to check the possibilities of assigning an item i (A item) to a set of m items in B and pick the item in B (j^*) that result in the closest value to the nominal value and that falls between the upper and lower specifications (UL, LL). The m items are chosen to be around the item (j) that would have been chosen if we solve for the Dynamic Variation Reduction (DVR) according to the rules of sorting. We check for m=1,…,P and then choose m*

that results in the largest throughput. The procedure for two subassembly groups with additive assembly function ($C=A+B$) is explained summarized in steps in Figures 6.2 and 6.3. Notice that if $i \geq P+1-\lfloor m/2 \rfloor^-$ or $i \leq \lfloor m/2 \rfloor^-$, we check less than m items in group B with the i^{th} item in group A; where $[\ ]^-$ stands for round-down to the largest integer, e.g. $[8.7]^-=8$. This part is not shown in the procedure explained in Figure 6.3. From its name, this algorithm is greedy because when we choose item j^* in group B (*Step* 4) to be mated with an A item, we cannot change this decision in a later step. This algorithm can be further generalized; however there are shortcomings when it comes to conflicting and nonlinear assembly systems of the subassembly groups on the final quality characteristics. The flowcharts are made to contain more details in purpose. Therefore, and through the dissertation the associated step-by-step procedure will be less detailed compared to the flowcharts.

6.3.2. Simulated Annealing Algorithms (SADTM 1 and SADTM 2)

The solution (S) for G number subassembly groups with P items in each group can be represented as shown in Equation 6.1, where s_1 represents an item from the first subassembly group for the first final assembly (part). The size of the vector is $P.G$. The first G items in S represent part 1, the next G items represent part 2, etc. Next, we explain the two SA algorithms in Sections 6.3.2.1 and 6.3.2.2.

$$S = \left\{ \underbrace{s_1\ s_2\ s_3\ \ldots s_G}_{\text{Part 1}};\ \underbrace{s_{G+1}\ s_{G+2}\ s_{G+3}\ \ldots s_{2G}}_{\text{Part 2}};\ \ldots;\ \underbrace{s_{(P-1).G+1}\ \ldots s_{P.G}}_{\text{Part } P} \right\} \tag{6.1}$$

6.3.2.1. SADTM 1 (Address-Based)

This algorithm is explained through a flowchart and a step-by-step procedure in Figures 6.4 and 6.5; respectively. We explain the algorithm here for the simplest case; when we have only two subassembly groups (A and B). At the beginning, the Solution is represented as follows: $S_1 = \{1\, b_1\ 2\, b_2\ \ldots P\, b_P\}$; this means that we assign the first item from A with b_1 from B, the second item from A with b_2, etc.

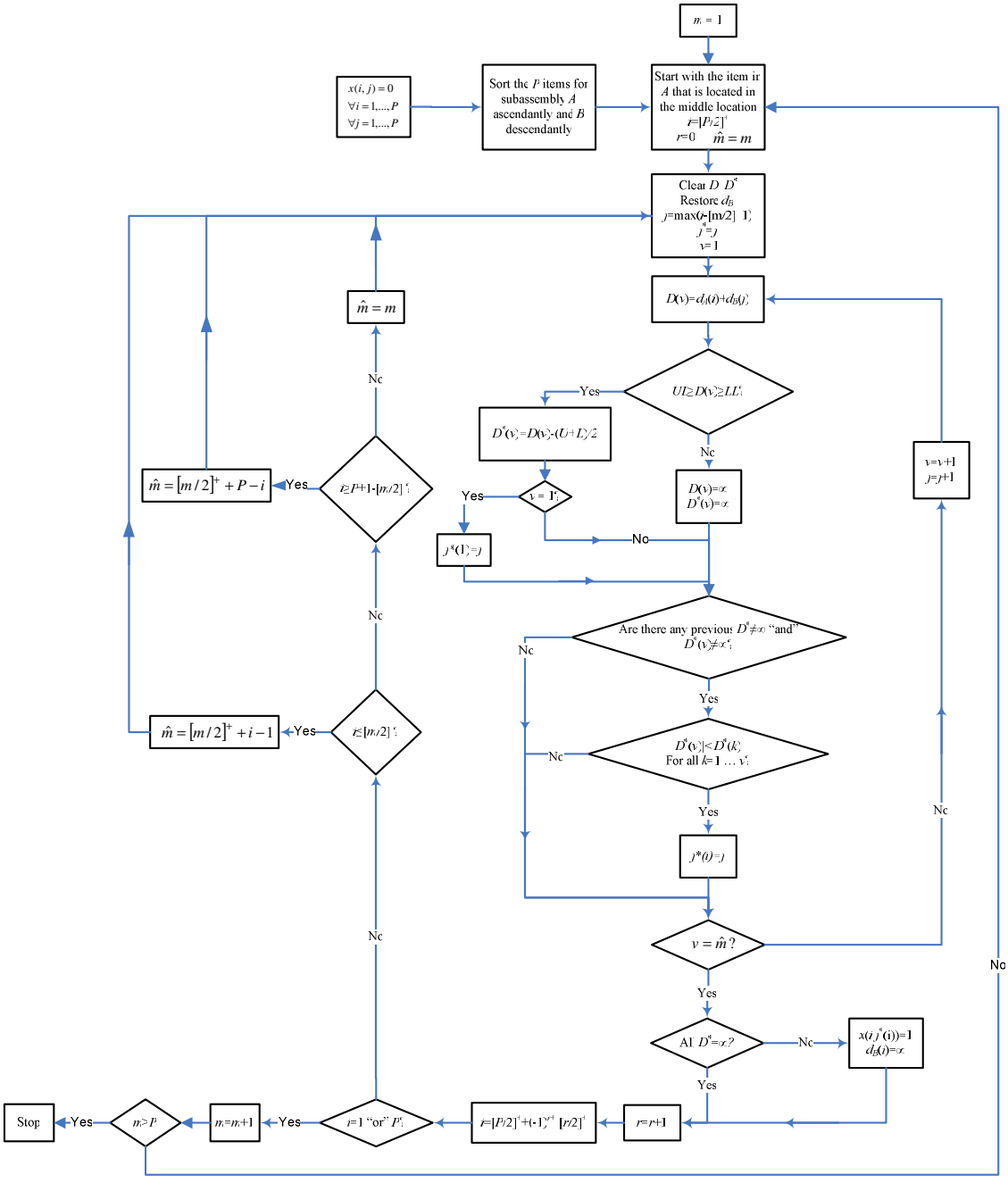


Figure 6-2. Flowchart for the greedy sorting heuristic to solve DTM problem with two subassembly groups (A and B) with additive assembly function: $C=A+B$

We assign random addresses (from a uniform probability density function between 0 and 1; U(0,1)) for each part: b_1, b_2, …, b_P. Then, we descendantly sort the b's according to the assigned addresses. We find the throughput of the initially generated solution. After that, we generate a neighboring solution according to our *exchange strategy* for SADTM 1. We accept all improving solutions; i.e. Throughput(New)≥Throughput(Old). We also accept a non-improving solution with a high probability at the beginning that diminishes with time by decreasing the annealing temperature according to a cooling scheme. The probability of accepting a solution in a maximization problem is given as follows:

Step 1	Sort the groups (*A* and *B*) in a descendant and ascendant orders; respectively
Step 2	$m = 1$ (Number of items in *B* to be checked with the i^{th} item in *A*)
Step 3	Start with the i^{th} item in *A* that is located in the middle of the group; $i = [P/2]^+$; where $[\ ]^+$ stands for ceiling of the value inside the bracket, e.g. $[3.3\ ]^+ = 4$. $r = 0$
Step 4	For item *i* in group *A*, find item *j* in a set of *m* items in group *B* that when assembled with i^{th} item in A, it gives a product with the *closet* value to the nominal. The *m* items are chosen to be around the item (*j*) that would have been chosen if we solve for the Dynamic Variation Reduction (DVR) according to the rules of sorting.
Step 5	Assign *infinity* to the chosen item in *B* so it will not be chosen for other *A* items; $d_{Bj^*1} \leftarrow \infty$.
Step 6	(a) $r = r + 1$ (b) Bounce up/down from the chosen *A* item (*i*). $i = [P/2]^+ + (-1)^{r+1}[r/2]^+$
Step 7	If $i = 1$ or *P* (all items assigned), go to *Step* 8; otherwise go back to *Step* 4
Step 8	Find the *Throughput* when we use *m* items; *Throughput* (*m*)
Step 9	$m=m+1$, reset the values for d_j's
Step 10	If $m = P$, go to *Step* 11; otherwise go to *Step* 3
Step 11	Find m^* that results in the highest throughput
Step 12	Using m^*, solve from *Step* 4 to *Step* 8 to find the best assignment.

Figure 6-3. Procedure of *Greedy Sorting Heuristic* to solve DTM problem with two subassembly groups (*A* and *B*) with additive assembly function: *C=A+B*

To generate a new neighboring solution, we generate new addresses for β (between 0 and 100%) of the P products. We experimentally found that we can obtain better results if the values of β were taken between 5% to 30%.

For a given annealing schedule, $Temperature = \{t_1, t_2, ...\}$, the algorithm goes as follows:

Step 1 (a) *Epoch* = 1.

(b) Generate initial sequence: $S_1 = \{1\, b_1\ 2\, b_2\ ...\ P\, b_P\}$.

Step 2 Estimate the achieved *throughput* for the initial solution S_1: T_{Epoch}

Step 3 Generate addresses randomly from U(0,1) for each part in the sequence, whereas Address(part$_i$)≥Address(part$_{i+1}$) by sorting

Step 4 Generate a temporary sequence S^* - Randomly generate new addresses for randomly chosen β (by generating a uniform random variable between 0 and 1 and comparing it with β) of the P parts in S. Sort items in subassembly B according to the new addresses; this will generate the S^* sequence.

Step 5 Estimate the achieved *throughput* for the temporary sequence S^* ; T^*

Step 6 If $\Delta T = T^* - T_{Epoch} \le 0$ (non-improving solution), continue; otherwise (improving solution) go to *Step* 9

Step 7 Generate a random variable k from U(0,1)

Step 8 If $k \ge e^{\Delta T / t_{Epoch}}$, (Do not accept non-improving solution): *Epoch* = *Epoch* +1, S(*Epoch*)=S(*Epoch*-1), T(*Epoch*)=T(*Epoch*-1), then go to Step 4;
otherwise continue

Step 9 (a) *Epoch*=*Epoch*+1

(b) S_{Epoch}=S^* and T_{Epoch}=T^*

Step 10 If stopping criterion is met (e.g. maximum number of Epochs), go to *Step* 4; otherwise STOP.

Figure 6-4. Procedure for SADTM 1 for two subassembly groups (G=2)

6.3.2.2. SADTM 2

The algorithm is performed as follows: after generating initial solution, we estimate the throughput achieved. After that we determine a neighboring solution by modifying the solution of β of the B items. The algorithm is further explained in Figures 6.6 and 6.7. Notice that we directly change the solution itself; rather than changing the addresses of the parts as we did in SADTM 1 (address-based algorithm).

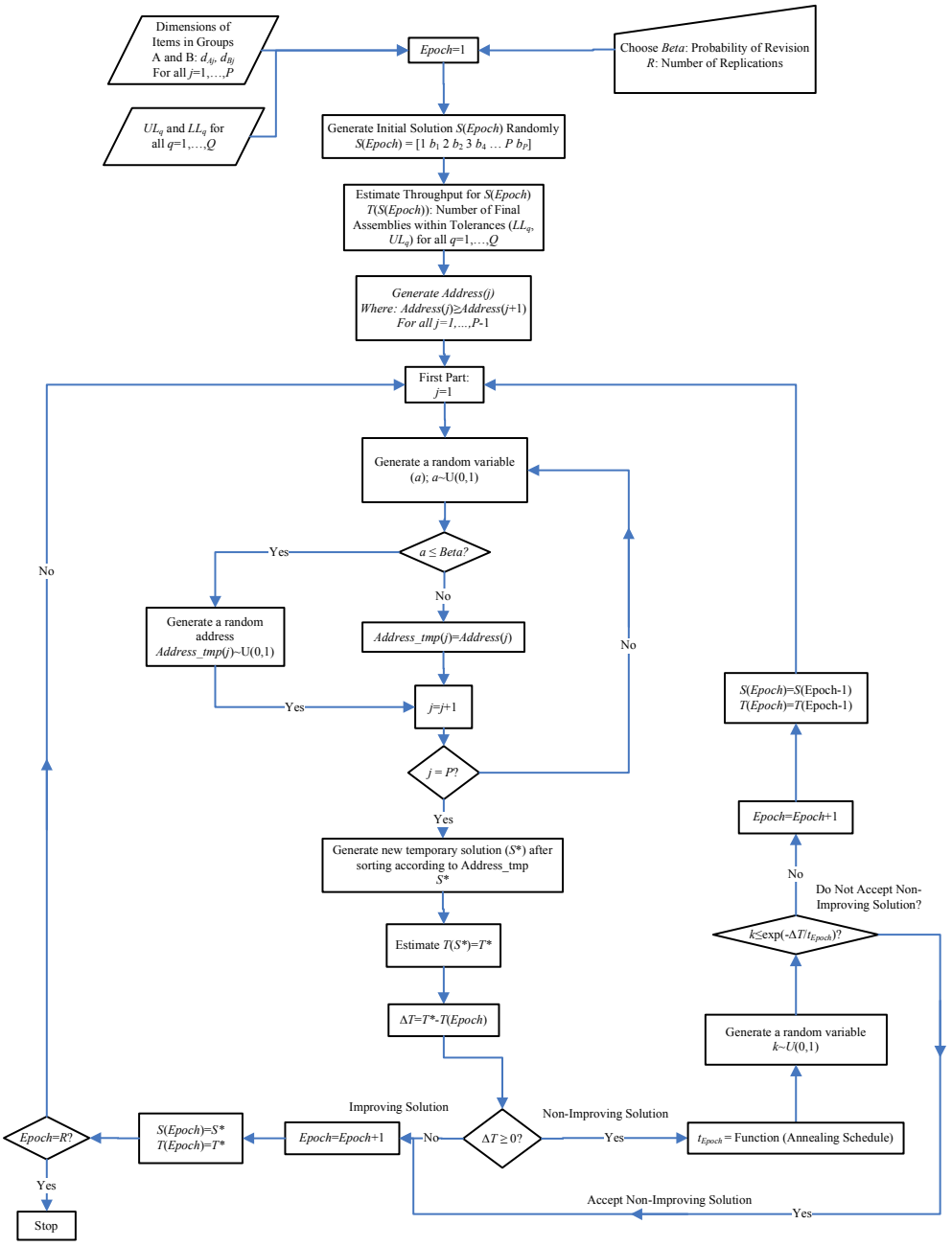


Figure 6-5. Flowchart for SADTM 1 for two subassembly groups (G=2)

For a given annealing schedule, $Temperature = \{t_1, t_2, ...\}$, the algorithm goes as follows:

Step 1 (a) *Epoch* = 1

(b) Generate initial sequence (solution):

$S_1 = \{1\ s_2\ s_3, ..., s_G; 2\ s_{G+2}\ s_{G+3}, ..., s_{2G}; ...; P, ..., s_{P.G}\}$.

Step 2 Estimate the achieved *throughput* for the initial solution S_1: T_{Epoch}

Step 3 Generate a temporary, neighboring solution S^* -

(a) Set i= 2, p = 1 (first part)

(b) Generate a random variable a from U(0,1)

(c) If $a \leq \beta$, re-generate subassembly items for part i: $S'_i = \{s'_i\ s'_{i+1}, ..., s'_{i+G-2}\}$; otherwise $S'_i = \{s_i\ s_{i+1},\ ..., s_{i+G-2}\}$

(d) If $p > 1$, continue; otherwise skip to *Step* 3(i)

(e) $j = i$

(f) If $s'_j \in \{..., s'_{j-2G}, s'_{j-G}\}$, re-generate new $s'_j \notin \{..., s'_{j-2G}, s'_{j-G}\}$; otherwise continue

(g) $j = j + 1$

(h) If $j \leq i+G-2$, go to *Step* 3(f); otherwise continue

(i) $i = i + G$, $p = p + 1$

(j) If $i \leq P.G$, go to *Step* 3(b); otherwise continue

(k) $S^* = \{1\ S'_1\} \cup \{2\ S'_2\} .. \cup \{P\ S'_{P.G}\}$

Step 4 Estimate the achieved *throughput* for the temporary sequence S^*: T^*

Step 5 If $\Delta T = T^* - T_{Epoch} \leq 0$ (non-improving solution), go to *Step* 7; otherwise (improving solution) go to *Step* 9

Step 6 Generate a random variable b from U(0,1)

Step 7 If $b \geq e^{-\Delta T / t_{Epoch}}$ (Do not accept non-improving solution): *Epoch* = *Epoch* +1, *S*(*Epoch*)=*S*(*Epoch*-1), *T*(*Epoch*)=*T*(*Epoch*-1), then go to Step 3; otherwise continue

Step 8 (a) *Epoch*=*Epoch*+1

(b) $S_{Epoch}=S^*$ and $T_{Epoch}=T^*$

Step 9 If the number of Epochs is less than *P*, go to Step 3; otherwise STOP.

Figure 6-6. SADTM 2 Algorithm for *G* Subassembly Groups

6.3.2.3. SA Parameter Control

One important parameter to control that can help enhancing our SA solution is the *annealing schedule* (or the *cooling scheme*). We start with a certain initial temperature; t_1. The probability of accepting a non-improving solution is given by:

$$\Pr = \min\left(e^{\frac{\Delta T}{t}}, 1 \right).100\% \qquad (6.2)$$

Notice that if $\Delta T \geq 0$, then we will certainly (100%) accept the solution because it is an improving solution. If the initial temperature is high, then we will start with a solution relatively similar to random search because we will be accepting non-improving solution ($\Delta T_i \leq 0$) more often. Accepting non-improving solution at certain stage in our search enables us to (hopefully) jump over some local optimal solutions as we progress with our annealing. On the other hand, if the initial temperature is small, then we will give less chance for non-improving solution to be accepted. Moreover, we will most probably behave similarly as a steepest descent solution. As we progress in our solution, we steadily decrease our temperature (cool) as follows:

$$t_{i+1} = 0.9.t_i \qquad (6.3)$$

We found it helpful to *reheat* the temperature at certain stage to a temperature less than the previous highest one. An example about that is shown in Figure 6.9. The small reheats are expected to jump over small local minima.

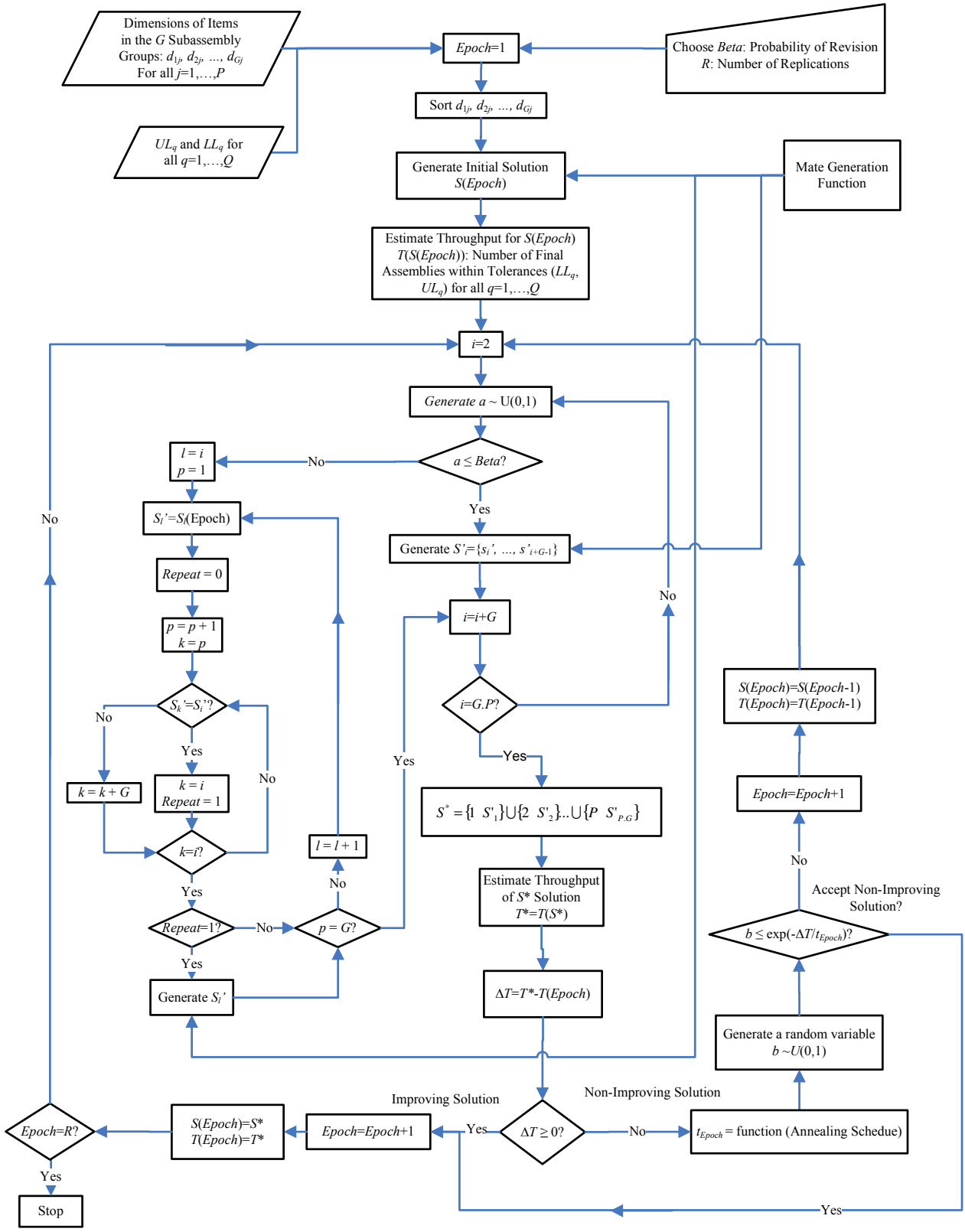


Figure 6-7. SADTM 2 with G subassembly groups

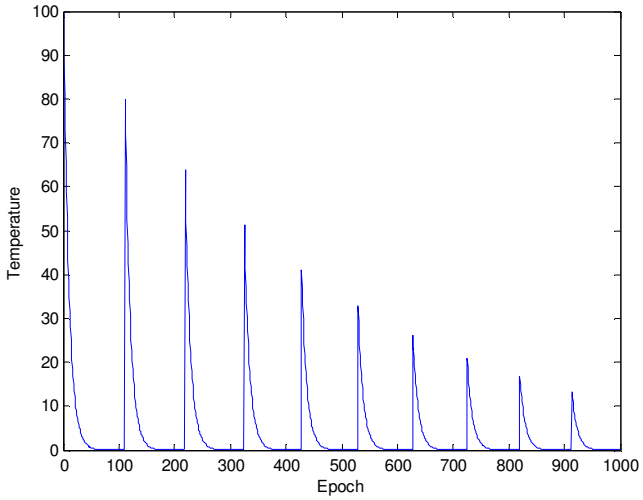


Figure 6-8. Temperature reheat schedule

Choosing the initial temperature is critical in SA. The approach we used to find our initial temperature was as follows: Generate n (say 30) random solutions and find their fitnesses (throughputs: T). The initial temperature is found as follows:

$$t_1 = \tau.(\max(T) - \min(T)) \qquad (6.4)$$

Suppose we choose τ = 2 and $\eta = \max(T_1, T_2, ..., T_n) - \min(T_1, T_2, ..., T_n)$, then we know that $|\Delta T| \le \eta$. Therefore, for a non-improving solution, the probability of accepting a non-improving solution is at least $e^{\Delta T / t_1} \ge e^{-\eta/2\eta} = 0.607$. If we decide our next temperatures to be annealed according to the schedule shown in Equation (9) with α=0.90, then we will accept the upcoming non-improving solutions with the following probabilities: 57.4%, 53.9%, 50.4%, 46.7%, etc.

We also found that controlling β could affect the solution considerably. It refers to the probability of changing a part (or its address) in the solution (sequence). For example, β = 5% means that we change 5 addresses or parts (on average) in a sequence that contains 100 parts in the sequence. Increasing β is expected to enhance the solution; however the computation time

increases dramatically by doing so. We have found that changing at least one part (or its address) in the sequence gives good results in a relatively short time.

6.3.3. Ant Colony Optimization Algorithms (ACODTM 1 and ACODTM 2)

In the following subsections, we explain the details about our ACO algorithms.

6.3.3.1. ACO Initialization

6.3.3.1.1. Graph Representation

As it was explained in Chapter 1, problems solved using ACO has to be defined in a *connected graph*. We define our graph as shown in Figures 6.10 and 6.11. A final assembly is built when an ant moves forward (first column to the G^{th} column) and another final assembly is built when it moves backward (G^{th} column to the first column). Ant movement in Figure 6.10 goes as follows (Figure 6.12): Forward → Jump inside the G^{th} column → Backward → Jump inside the first column → Forward → … etc. Alternatively, it can go as follows (Figures 6.11 and 6.13): Forward → Jump Backward (from the G^{th} column to the first column)→ Forward → Jump Backward (from the G^{th} column to the first column)… etc. Each completed tour by an ant produces a set of P final assemblies. Another scenario is to move from a subassembly group to another randomly until all the P final assemblies are made.

6.3.3.1.2. Pheromone Initial Quantity

At the beginning, we virtually deposit the same amount of pheromone (τ_{gij}) in all the edges. Pheromone amount (τ_{gij}) refers to the amount of pheromone in the edge that links (g, i) and (g+1, j) nodes; which respectively represent the i^{th} item in group g and the j^{th} item in group g+1. The amount of deposited pheromone at this initialization stage impacts the solution quality. We will discuss this issue later in this Chapter.

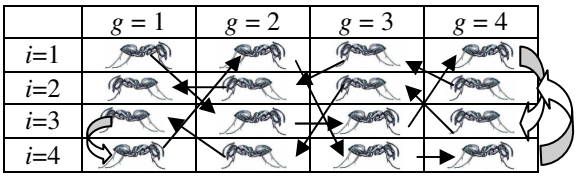


Figure 6-9. Graph representation 1 for a virtual ant tour; four subassembly groups and four final assemblies

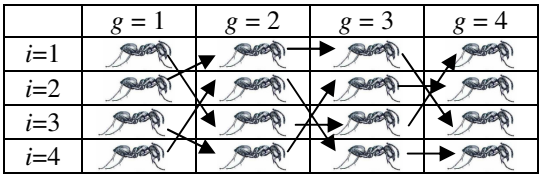


Figure 6-10. Graph representation 2 for a virtual ant tour; four subassembly groups and four final assemblies

1	3	3	1	3	4	2	3	4	1	4	4	2	2	1	2
Part 1				Part 2				Part 3				Part 4			

Figure 6-11. Ant tour for the graph shown in Figure 6.10

1	3	3	1	2	1	1	3	3	4	2	2	4	2	4	4
Part 1				Part 2				Part 3				Part 4			

Figure 6-12. Ant tour for the graph shown in Figure 6.11

6.3.3.1.3. Virtual Ant Placement

Initially, we place the *m* ants randomly or heuristically in different nodes in the graph. In this Chapter, we will place them randomly and uniformly to study the effects of both strategies.

6.3.3.2. ACO Tour Construction

Ant moves from one node to another in its tour according to two factors: *pheromone amount* and *visibility* (*coordination rate*). The visibility (η_{gij}) is essentially determined by a *greedy heuristic*. For example, in *Traveling Salesman Problem*, it is the reciprocal of the distance between a city

and another one. Referring to Figure 6.14, we combine the two factors effect by evaluating the probability of moving from node (g, i) to node (g+1, j) for ant k. The probability is given as follows:

$$P_{gij}^{\ k} = \frac{(\tau_{gij})^{\alpha}(\eta_{gij})^{\beta}}{\sum_{l \in N_i^{\ k}} (\tau_{gil})^{\alpha}(\eta_{gil})^{\beta}} \qquad (6.5)$$

Where:

$N_i^{\ k}$: Set of nodes (items in group g+1) to be visited by ant k that is currently located in item i in group g.

α, β: exponents that determine the effects of pheromone amount and visibility on the probability; respectively.

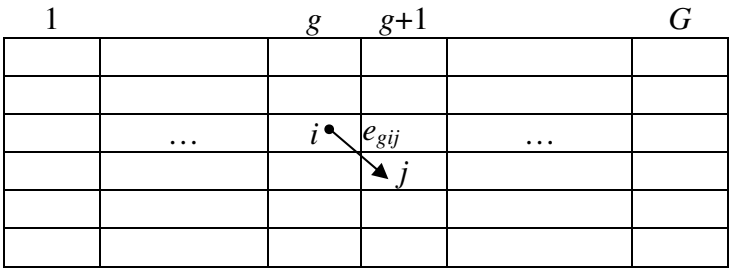


Figure 6-13. General graph representation for DTM problem

6.3.3.2.1. Greedy Heuristic to estimate η

We will mainly take advantage of using a *greedy heuristic* in this algorithm when we are about to assemble the G^{th} item in a final assembly. The accumulated quantity for the q^{th} quality characteristic up to the $(G\text{-}1)^{\text{th}}$ assembled item for the k^{th} ant is denoted by: $\hat{y}_q^{\ k}$. Now, we can estimate the final quality characteristics if we would have assembled the previously assembled items with the j^{th} item in G^{th} group. This can be estimated as follows:

$$y_{qj}^{\ k} = \hat{y}_q^{\ k} + d_{Gjq} \qquad (6.6)$$

The *visibility* ($\eta_{G-1,ij}$)is estimated as follows:

$$\eta_{G-1,ij} = \begin{cases} \Gamma & if\ LL_q \le y_{qj}{}^k \le UL_q \quad \forall q = 1,...,Q \\ \gamma & Otherwise \end{cases} \qquad (6.7)$$

We typically choose Γ and γ to be 1 and 0.2; respectively.

6.3.3.3. ACO Pheromone Update

After the ants complete their tours, the pheromone amounts in all the links (g, i, j) are updated according to the following Equation:

$$\tau_{gij} = (1-\rho)\tau_{gij} + \sum_{k=1}^{m} \Delta\tau_{gij}{}^k \qquad \forall g = 1,..., G-1, \forall i = 1,..., P, \forall j = 1,..., P \qquad (6.8)$$

Where:

$$\Delta\tau_{gij}{}^k = \begin{cases} Throughput^k & if\ arc\ (g,i,j)\ is\ used\ by\ ant\ k \\ 0 & Otherwise \end{cases}$$

ρ : Pheromone evaporation rate

$Throughput^k$: Throughput achieved when items are assembled according to the k^{th} ant tour.

6.3.3.4. ACODTM 1 Algorithm

The detailed algorithm is depicted in the flowchart in Figures 6.15 and 6.16. This procedure is close to *Ant Systems* introduced in [Dorigo et al. (1996)] for solving the Traveling Salesman Problem (TSP).

6.3.3.5. ACODTM 2 Algorithm

This algorithm differs from ACODTM 1 in three-folds. First, we use a 3-opt local search after an ant k completes its path to enhance the solution. Second, we have modified the *state transition*

rule as shown in Equation 6.9 where v refers to a randomly generated variable from a uniform distribution between 0 and 1 and v_0 refers to the percentage of the exploited paths. This is the version of the state transition rule that was proposed for Ant Colony System (ACS) presented in [Dorigo and Stützle (2004)]. Third, we generated the paths to make the parts randomly from a subassembly group to another. The flowchart for the algorithm is shown in details in Figure 6.16.

$$P_{gij}^{\ k} = \begin{cases} \arg\max_{l \in N_i^k} \left((\tau_{gil})^{\alpha} (\eta_{gil})^{\beta} \right) & \text{if } v \le v_0 \quad (\textit{Exploitation}) \\ \dfrac{(\tau_{gij})^{\alpha} (\eta_{gij})^{\beta}}{\sum_{l \in N_i^k} (\tau_{gil})^{\alpha} (\eta_{gil})^{\beta}} & \text{otherwise} \quad (\textit{Exploration}) \end{cases} \tag{6.9}$$

6.3.4. Comparison between the Algorithms

Comparison between the algorithms described in this Chapter is summarized in Table 6.1.

Table 6.1. Qualitative comparison between Greedy, SADTM and ACODTM algorithms

Algorithm	Advantages	Disadvantages
Greedy	- Very Quick Solution - Easy to implement - No parameters to control	- Applicable for non-conflicting assembly system with additive assembly systems
SADTM 1	- Relatively easy to implement - Few parameters to control	- Very slow
SADTM 2	- Relatively easy to implement - Best for non-conflicting assembly system with additive assembly functions - Few parameters to control	- Slow
ACODTM 1	- Fast -Best applicable for two subassembly groups.	- Relatively not easy to implement - Many parameters to control.
ACODTM 2	- Very fast - Best for large problems	- Relatively not easy to implement - Too many parameters to control.

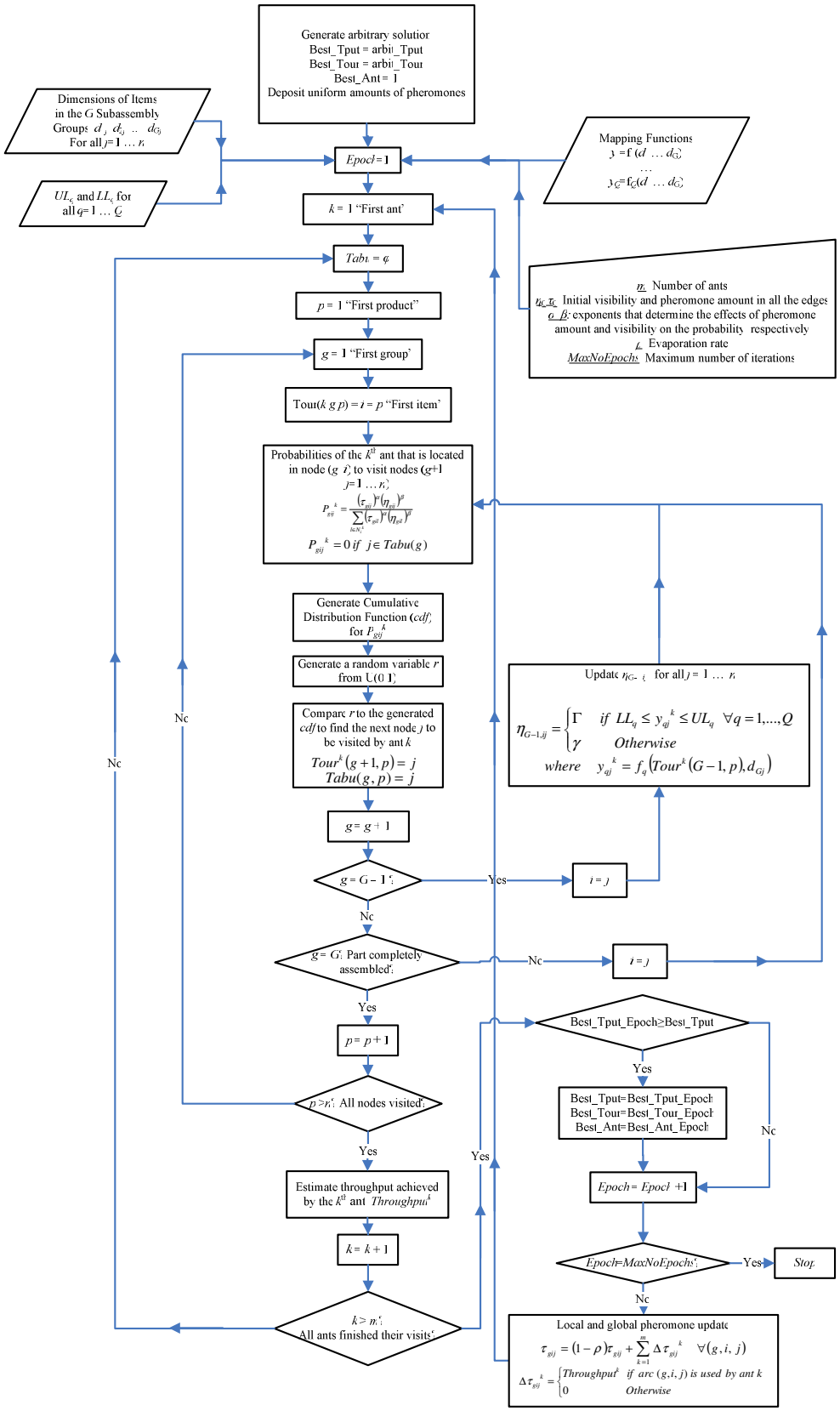


Figure 6-14. ACODTM 1 algorithm

Step 1	Generate arbitrary solution and find the achieved *throughput;* T_{Best}
Step 2	Deposit a uniform amount of pheromone and visibilities in the edges
Step 3	*Epoch* = 1
Step 4	Set k = 1 (first ant)
Step 5	$\text{Tabu}^k(g) = \{\}$ $\quad \forall g = 1,...,G$
Step 6	p = 1 (first part)
Step 7	g = 1 (first group)
Step 8	Tour(k,g,p) = p = i
Step 9	Estimate the probabilities of the k^{th} ant that is located in node (g, i) to visit nodes (g+1, j=1,…,n) according to Equation (11).
Step 10	Generate Cumulative Distribution Function (*cdf*) for $P_{gij}^{\ k}$
Step 11	Generate a random variable r from U(0,1)
Step 12	Compare r to the generated *cdf* to find the next node (g+1, j) to be visited by ant k. $Tour^k(g+1, p) = j$ Include the selected item in the Tabu list. $Tabu^k(g) = j$
Step 13	$g = g + 1$
Step 14	$i = j$
Step 15	If $g = G$-1, find $\eta_{G-1,ij}$ according to Equations (12) and (13) and then go back to *Step* 9
Step 16	If $g = G$, go back to *Step* 9
Step 17	$p = p + 1$
Step 18	If $p > n$ (all nodes visited by ant k), go back to Step 7
Step 19	Estimate the *throughput* achieved by ant k; *throughput*(k)
Step 20	$k = k + 1$
Step 21	If $k < M$ (all ants constructed their tours)*, go to Step 5;* otherwise continue
Step 22	If $T_{Best} \leq$ max(*throughput*(1), *throughput*(2), …, *throughput*(M)), continue; otherwise skip to *Step* 24
Step 23	T_{Best} = max(*throughput*(1), *throughput*(2), …, *throughput*(M)) *Best_Ant* = Ant that achieves the best *throughput* *Best_Tour* = Tour(*Best_Ant*)
Step 24	*Epoch* = *Epoch* + 1
Step 25	If a stopping criterion is met, STOP; otherwise continue
Step 26	Update the pheromones according to Equation (14) and then go back to *STEP* 4

Figure 6-15: ACODTM 1 Procedure

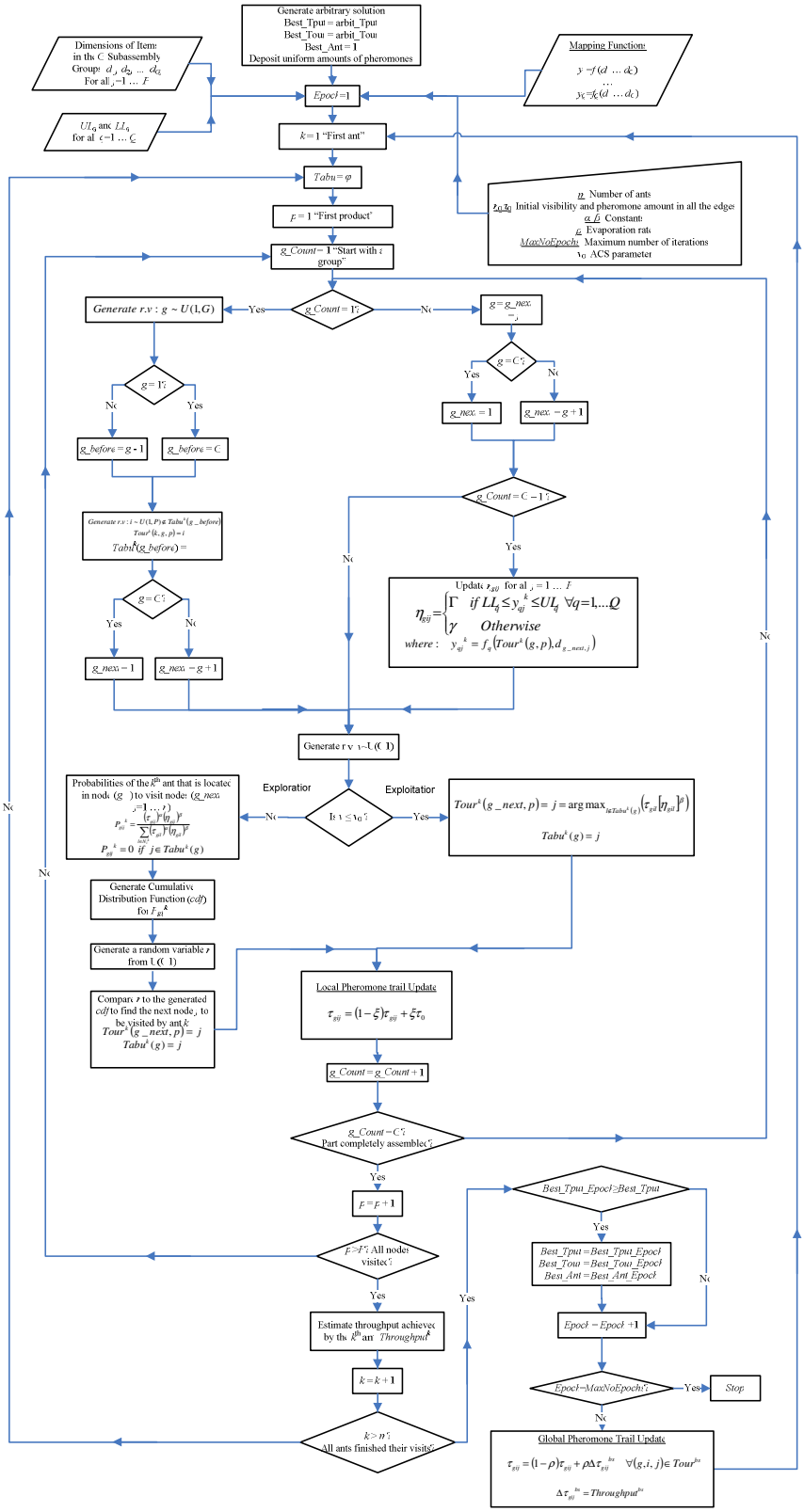


Figure 6-16. ACODTM 2 algorithm

6.4. NUMERICAL EXAMPLES

Example 1: Given that we have two subassembly groups *a* and *b* (Table 6.2), we want to maximize the throughout for the final requirement (*c*); where $c = a + b$ and $UL_C = 20.2$ and $LL_C = 19.8$.

Table 6.2. Data for Example 1

No.	*a*	*b*	*No.*	*a*	*b*	*No.*	*a*	*b*	*No.*	*a*	*b*
1	9.433384812	8.76281864	10	10.01132154	9.499225438	19	9.754061848	11.04918572	28	8.328188189	10.36654693
2	8.259796789	11.07300144	11	9.667390776	9.593398332	20	9.05286978	10.45160289	29	8.473590054	9.890991857
3	9.964391289	9.982648894	12	10.53590744	11.42521171	21	10.12532022	10.83401071	30	10.38871831	10.94215051
4	10.11891042	10.5955778	13	9.285140314	10.39878641	22	8.573322108	9.51231487	31	9.464488907	10.52621304
5	8.753888568	10.52797431	14	11.92305626	9.476453505	23	10.52499411	10.02469155	32	10.5018395	9.641319118
6	10.97861334	10.37395222	15	9.715278398	10.32853094	24	11.39040637	10.16826225	33	10.62140929	11.32892984
7	10.97694649	9.631042234	16	9.953539824	9.54091487	25	9.186667887	9.479471326	34	10.52269434	10.56814422
8	9.809280647	9.672192676	17	10.86044998	9.38084842	26	10.66174123	9.402467895	35	11.07315972	11.42987541
9	10.15661687	10.08712605	18	9.901524094	10.16923187	27	11.03865855	11.03306709	36	10.48147396	11.1829013

We solved the problem using the two SA algorithms with β=0.10 and the results are reported in Figure 6.17. We also solved the problem using our ACODTM 1 algorithm and the results are shown in Figure 6.18 and Table 6.3. All the implementations were using MATLAB on a 2.4 GHz, Pentium 4, 512 MB processor. The greedy algorithm was used to solve the problem as well and the results were found to be identical to ACODTM 1 algorithm (Figure 6.18). We can notice that ACODTM 1 Algorithm solution outperforms SADTM solutions in terms of the found throughput value and the computation time. Notice in the lower part of Figure 6.18, we notice that all the ants converge to the same solution eventually; which gives an indication of how the algorithm is performing. On the other hand, SADTM 2 outperforms SADTM 1 (Figure 6.17) because it used the modified sorting heuristic which fits well in this example. In Figure 6.19, the results for the same problem using the greedy algorithm is reported. We notice that the solution was found to be the same as ACODTM 1 but with less computation time. However, the greedy heuristic is only applicable here because we had a very simple case (non-conflicting, linear assembly systems).

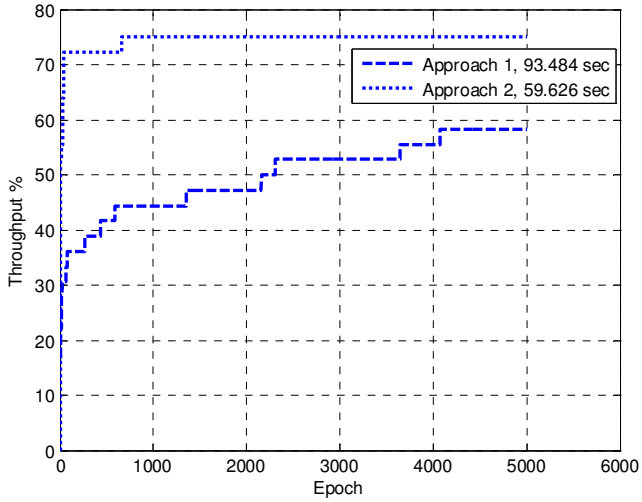


Figure 6-17: SADTM 1 and 2 solutions with β=0.10; Throughput for SADTM 1 and 2: 59% and 75%; respectively. Computational time are shown in the legend

Table 6.3. Solution using ACODTM 1 algorithm

a	***b***	***a***	***b***	***a***	***b***	***a***	***b***
18	18	1	6	8	15	2	2
10	9	33	22	4	3	14	34
9	8	12	7	7	17	20	27
3	29	32	14	28	5	15	13
16	24	26	26	21	23	19	28
11	20	36	16	24	1	27	12
29	35	6	21	30	25	17	10
31	4	5	19	25	30	22	33
13	31	23	32	34	11	35	36

Table 6.4. Effect of changing α and β on ACODTM 1 algorithm

α	β											
	0		1		2		3		4		5	
	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ
0	0.3222	0.0248	0.55	0.0232	0.6944	0.0196	0.7444	0.0124	0.75	0	0.7611	0.0152
1	0.3778	0.0248	0.7611	0.0152	**0.7778**	**0**	**0.7778**	**0**	**0.7778**	**0**	**0.7778**	**0**
2	0.3444	0.0751	0.7333	0.0317	0.7611	0.0152	**0.7778**	**0**	**0.7778**	**0**	**0.7778**	**0**
3	0.3056	0.034	0.7333	0.0152	0.7667	0.0152	0.7667	0.0152	0.7722	0.0124	0.7722	0.0124
4	0.3	0.0362	0.7	0.0232	0.7556	0.0124	0.7667	0.0152	0.7778	0	0.7778	0
5	0.3056	0.0196	0.7278	0.0232	0.7556	0.0124	0.7722	0.0124	0.7722	0.0124	0.7722	0.0124

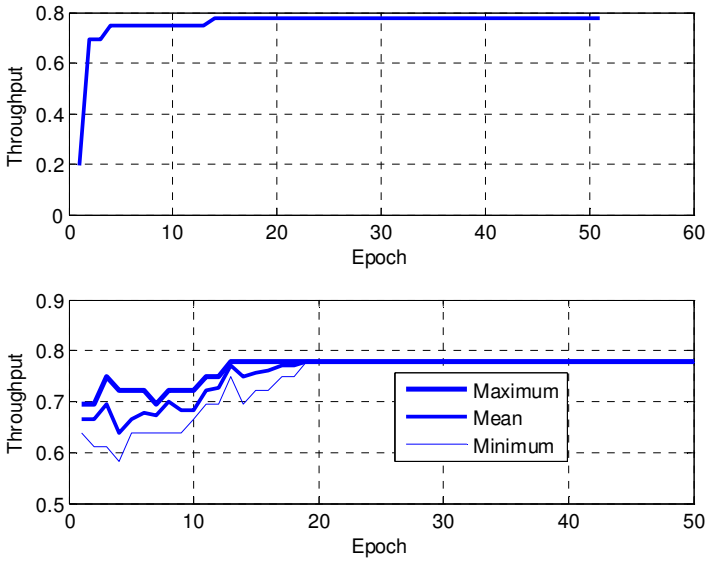


Figure 6-18. ACODTM 1 solution; with 5 ants (m=5), ρ=0.5, α=3 and β=5; Execution time: 34.46 seconds; Throughput=77.8%

We determined α and β for ACODTM 1 by solving for different values as it is shown in Table 6.4. Then we solved using different *numbers of ants* as it is shown in figure 6.20. Finally, the *evaporation rate* was determined by solving for a range between 0 and 1 (Figure 6.21). The best parameters found for α, β, m and ρ are 5, 2, 4 and 0.2; respectively.

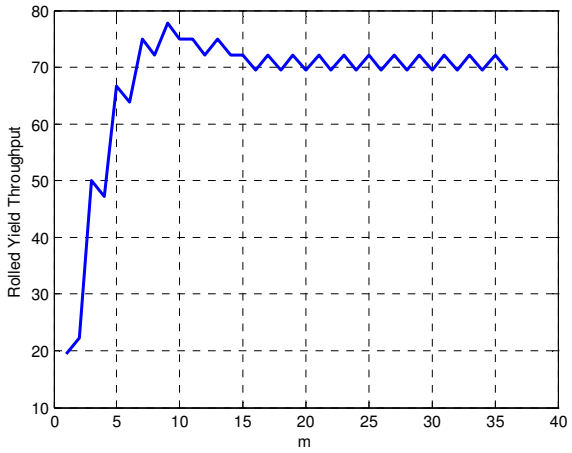


Figure 6-19. Greedy algorithm; throughput = 77.8% at m^* = 9; Execution time: 0.52 seconds

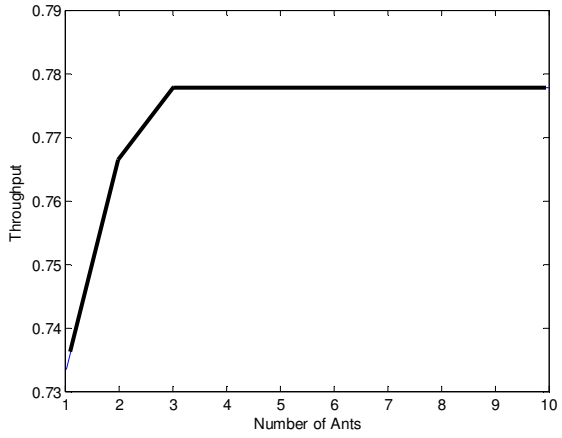


Figure 6-20. Effect of changing the number of ants (m) on the throughput

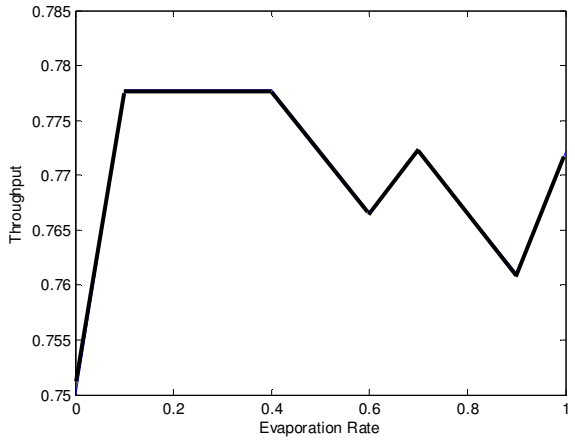


Figure 6-21. Effect of changing the evaporation rate (ρ) on the throughput

Example 2 (Comparison between SADTM 2 and ACODTM 1): We want to maximize the throughput of assembling 20 final assemblies from 4 subassembly groups (x_1, x_2, x_3 and x_4) so we can satisfy 3 final quality characteristics simultaneously (y_1, y_2 and y_3). The data for this example is the first 20 items in Table 6.5. Lower Limits for the 3 final quality characteristics are respectively: 85, 40 and 4 and the Upper Limits are respectively: 115, 50, and 5. The mapping functions between the x's and the y's are as follows:

$$y_1 = 100 - 50x_1 + x_2 - x_3 + 50x_4$$

$$y_2 = 35 - x_1 + x_2 - x_3 + 12x_4$$

$$y_3 = 2.5 - x_1 + x_2 + 2x_4$$

Table 6.5. Data for examples 2 and 3

No.	x_1	x_2	x_3	x_4	No.	x_1	x_2	x_3	x_4
1	1.313	0.9583	0.8645	0.8017	25	0.9369	0.9572	1.0502	0.8821
2	1.3741	1.1037	0.9953	1.0151	26	1.1543	0.9707	0.8558	1.037
3	1.0938	1.0048	0.8661	1.0488	27	1.4857	0.8032	1.1248	1.0038
4	1.2034	0.7776	1.0043	0.8166	28	1.0376	0.962	0.9025	0.8004
5	1.1723	0.847	1.0002	0.8652	29	0.9976	1.1671	0.9155	0.9526
6	1.2643	0.8634	1.0125	0.8797	30	0.9837	0.8834	1.0399	0.6777
7	1.4754	0.9416	0.8798	0.9009	31	1.0834	1.1014	0.953	0.9086
8	1.3834	0.8964	0.9557	0.8745	32	1.4772	0.7779	1.0902	1.0019
9	1.7238	0.8733	1.0177	1.1675	33	1.3024	1.0712	1.0655	1.0125
10	1.4881	0.9941	1.0566	0.8253	34	1.3526	1.1413	1.1751	0.8973
11	1.1326	0.814	1.1636	1.1978	35	0.9866	0.9299	0.9845	0.7814
12	1.2415	1.0686	1.06	0.9495	36	1.1266	1.0505	0.8552	0.9308
13	1.1499	1.0227	0.8666	0.941	37	1.2509	1.0247	1.1738	0.8314
14	1.4385	0.8945	0.8889	1.0046	38	1.1806	1.2506	0.9545	0.8043
15	1.2419	1.0241	1.1459	1.1874	39	1.2866	1.1489	0.9422	1.0191
16	1.5252	1.0657	0.8299	0.7642	40	1.4394	0.8917	1.0557	0.8828
17	1.6813	0.888	0.8306	0.8498	41	1.4313	0.9611	1.2174	0.7155
18	1.4243	0.9494	0.9407	1.1901	42	1.62	1.0291	1.2113	1.2059
19	1.2833	0.9763	1.0815	0.8922	43	1.1758	1.0064	0.9175	0.8994
20	1.3656	0.9909	0.9114	0.6903	44	1.3349	0.8124	0.8922	0.8572
21	1.0558	0.9387	0.8568	0.948	45	1.4341	1.0708	0.8758	0.4094
22	1.2853	0.9265	0.9621	1.1899	46	1.2562	1.1166	1.0859	0.6812
23	1.1914	0.7955	1.0861	1.0524	47	1.2388	1.1473	0.8507	1.0559
24	1.0243	1.0112	0.8169	1.0026					

Notice that the system is *non-conflicting*, yet solving it using the greedy algorithm is not quite obvious. Figure 6.22 presents the progress of the solution over iterations (epochs) for SADTM 2 with $\beta = 0.10$. It took more than 15 seconds to converge to 60% in a 2.4 GHz, 512 MB, and Pentium 4 processor. Additionally, we solved the same problem using our ACODTM 1 algorithm and the results are depicted in Figure 6.23. It took less than 15 minutes on the same processor to solve reach 65% throughput. We found that results from running the same codes

(SADTM 2 and ACODTM 1) for 30 seconds are consistent. This validates the conclusion that ACODTM 1 outperforms SADTM2 algorithm.

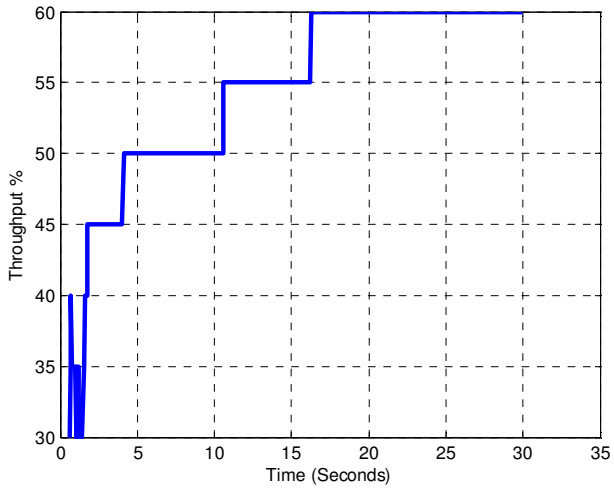


Figure 6-22. Example 2 solution using SADTM 2

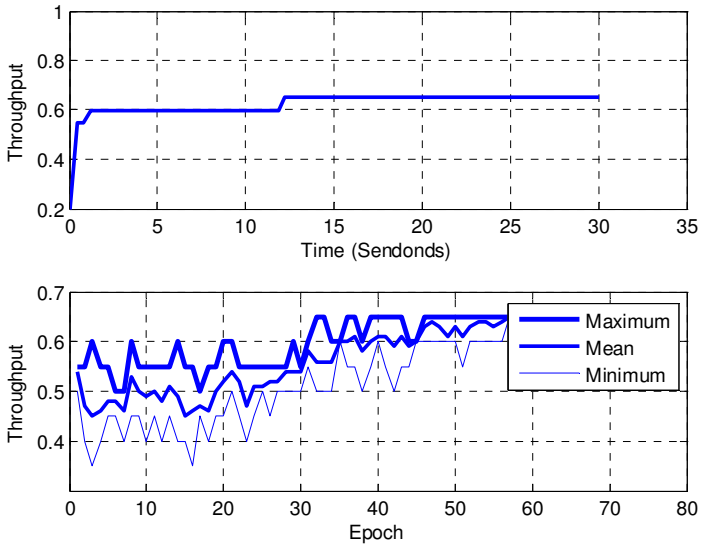


Figure 6-23. Example 2 solution using ACO1 algorithm; with 5 ants (m=5), ρ=0.2, α=3 and β=5

Example 3 (Comparison among SADTM 2, ACODTM 1 and ACODTM 2): We solve the same problem in Example 2 but with 47 items in each group instead of 20 (data is shown in Table 6.5). Table 6.6 summarizes the performance of SADTM 2, ACODTM 1 and ACODTM 2

in solving the problem. The performance is measured by the means and standard deviations of the found throughput values by running the code for 30 seconds. We notice that ACODTM 2 outperforms ACODTM 1 and SADTM 2 in solving the problem. The used parameters in all the cases are given in Table 6.7. Figure 6.24 reports the result we got when we ran ACODTM 2 code for 1 hour. We obtained a throughput of 76.60% in 500 seconds. The plot the in bottom of Figure 6.24 indicates that the ants are still searching for a better solution.

Table 6.6. Algorithm performances in 30 second runs

	SADTM 2	**ACODTM 1**	**ACODTM 2**
Mean	0.65534	0.67664	0.71914
SD	0.017821	0.009526	0.009526

Table 6.7: Algorithm parameters

SADTM 2	**ACODTM 1**	**ACODTM 2**
$\beta = 0.10$	$m = 10$ $\rho = 0.2$ $\alpha = 3$ $\beta = 5$	$m = 10$ $\rho = 0.1$ $\alpha = 3$ $\beta = 2$ $v_0 = 0.9$

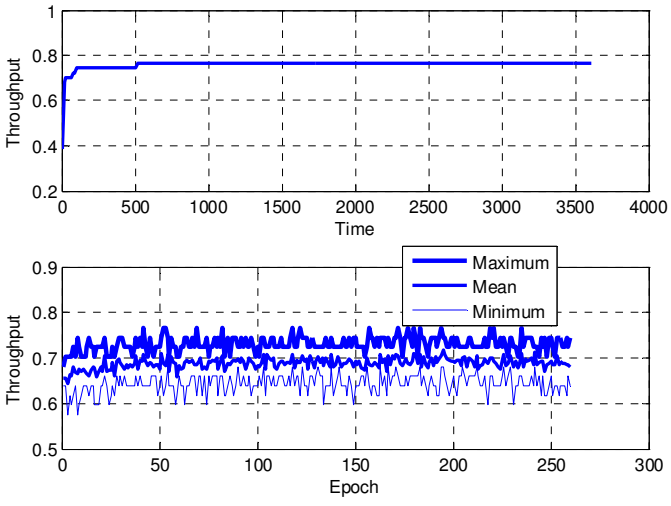


Figure 6-24. ACODTM 2 solution in 1 hour; 76.60%

6.5. SUMMARY

We reviewed the model of Dynamic Throughput Maximization (DTM) in this Chapter. We found that a problem instance with a practical size is hard to solve quickly, therefore we resorted to heuristics. The presented heuristics are: greedy heuristic that is based on the 3-Rule heuristic we presented in an earlier publication, two Simulated Annealing algorithms (SADTM 1 and SADTM 2) and two Ant Colony Optimization algorithms (ACODTM 1 and ACODTM 2). We found that the greedy heuristic is only valid for a non-conflicting system with additive assembly functions. We also found that the Ant Colony Optimization algorithms typically outperform those Simulated Annealing algorithms. More specifically, ACODTM 2 outperforms ACODTM 1 because of the use of 3-opt local search after fully constructing a tour by an ant and by modifying the state transition state according to the proposed Ant Colony System rule in [Dorigo and Stützle (2004)]. We also believe that the random generation of final products also contributed in the algorithm's success to find a solution quicker than the other algorithms. It is worth mentioning that this model and approach works well when we have different sizes of subassembly groups; for example: if we have 20 *a*'s, 30 *b*'s and 22 *c*'s in a 3-bar mechanism shown in Figure 6.1(c). This can be achieved by including some virtual subassemblies items with values that are far-off from the acceptable ranges.

CHAPTER 7: HYBRID IMPLEMENTATION OF INSPECTION PLANNING AND DYNAMIC THROUGHPUT MAXIMIZATION

7.1. INTRODUCTION

After inspecting a batch of subassemblies, we often have the choice of either assembling the inspected subassemblies arbitrarily or selectively. In this Chapter, we revise a previous model for inspection planning in agile production environment that we presented in Chapter 3 by integrating it with another previously presented work on Dynamic Rolled Yield Throughput Maximization (DTM) presented in Chapters 5 and 6. Integrating these models is expected to further minimize the cost of failure of the final assemblies and therefore the total cost. We present our approach that is mainly based on metaheuristic algorithms and Monte Carlo Simulation. We also solve a numerical example to compare the results with and without implementing the proposed integration.

The rest of the Chapter is organized as follows: we explain the problem background and its motivation in Section 7.2. In Section 7.3, we explain our approach for integrating inspection planning model with the Dynamic Throughput Maximization (DTM) model. Next, a numerical example is solved for illustrative and demonstrative purposes in Section 7.4. We compare the total costs when we assemble subassemblies arbitrarily after inspection and when we assemble them based on DTM. In Section 7.5, we conclude with the summary of this Chapter and some general remarks.

7.2. PROBLEM

As we introduced earlier in Chapter 3, inspection planning is the act of determining *what*, *when* and *how* to inspect features so we can minimize the total cost. We represented the total cost in an earlier Chapter based on a work for Chen and Thornton (1999) as: the cost of inspection, rework,

scrap of subassemblies and the cost of final assembly failure. We assume that inspection and assembly is executed in batches. We had five decision variables for this problem (for each subassembly group): frequency of inspection of a subassembly feature, and four limits (LL_S, LL_R, UL_R, UL_S) that determine when to keep, scrap or rework an inspected item (refer to Figure **3.**1). Usually, the collected data for the inspected features during inspection are merely used to assess whether the inspected part has to be *kept*, *reworked* or *scrapped*. Inspection planning requires using data to map the quality characteristics together. This can be obtained by using historical data for existing products. However, for newly-launched products we do not have this data. Therefore, we proposed using simulated data that can be generated from Variation Prediction CAD software such as 3DCS and VSA. After inspecting a batch of subassembly items, it makes sense to utilize this data to intelligently assemble subassembly items together to further reduce the variation of the final assemblies (or maximize the final throughput). Sorting the items and assembling them in this manner adds up in the assembly cost. Therefore, our focus in this work is to reach a trade-off between the assembly and failure costs. In this work, we aim at hybridly implementing inspection planning along with yield maximization models.

7.3. APPROACH

7.3.1. Inspection Planning with Dynamic Throughput Maximization (DTM)

The decision variables for the inspection planning problem is to optimally determine the following for each subassembly feature: (1) corrective plan limits (LL_S, LL_R, UL_R, UL_S), (2) frequency of inspection (*freq*) and (3) a binary variable that represents whether to implement Dynamic Rolled Yield Throughput Maximization (DTM) or not (τ_{DTM}), where τ_{DTM}=1 if we decide to implement DTM and 0 otherwise. Notice that the first and the second decision variables (5 decision variables for each subassembly) are the decision variables for the inspection planning that we presented in Chapter 3. The third decision variable is proposed in this work as we are seeking whether to implement DTM or not. The total cost (TC) we want to minimize is:

$$TC = C_I + C_S + C_R + C_F + C_{DTM} \qquad (\mathbf{7.}1)$$

Where:

C_I: Inspection cost of subassembly feature

C_S: Scrap cost of subassembly feature

C_R: Rework cost of subassembly feature

C_F: Failure cost of final assembly

C_{DTM}: Cost of implementing Dynamic Throughput Maximization (DTM)

The first four terms were proposed in Chen and Thornton (1999) to represent the cost of inspection. We propose to solve the problem using a Genetic Algorithm (GA). To solve this problem, we need to determine the following decision variables that minimize the total cost:

(1) LL_R for each subassembly group

(2) LL_S for each subassembly group

(3) UL_R for each subassembly group

(4) UL_S for each subassembly group

(5) Frequency (*freq*) of inspection for each subassembly group

(6) One binary variable that indicates whether it is better to implement DTM or not

The number of decision variables is: $5.M+1$; where M is the number of subassembly groups. Since the first five decision variables are continuous and GA solves only combinatorial optimization problems, we discretize those variables into different resolutions. Figure **7.**1 represents our chromosome representation that contains the decision variables. Figure **7.**2 explains how we generate offspring solutions from two parents by *crossing over* the genes in the parents. Unlike how chromosomes are represented in GA literature, we represent our chromosomes in matrix forms as it is show in Figure **7.**1.

$$\begin{array}{ccccc} LL_{R1} & LL_{R2} & \cdots & LL_{RM} & \tau_{\mathrm{DTM}} \\ LL_{S1} & LL_{S2} & \cdots & LL_{SM} & \tau_{\mathrm{DTM}} \\ UL_{R1} & UL_{R2} & \cdots & UL_{RM} & \tau_{\mathrm{DTM}} \\ UL_{S1} & UL_{S2} & \cdots & UL_{SM} & \tau_{\mathrm{DTM}} \\ freq_1 & freq_2 & \cdots & freq_{\mathrm{M}} & \tau_{\mathrm{DTM}} \end{array}$$

Figure 7-1. Chromosome representations of the decision variables. The $M+1^{\text{th}}$ column is the binary decision variable for implementing DTM

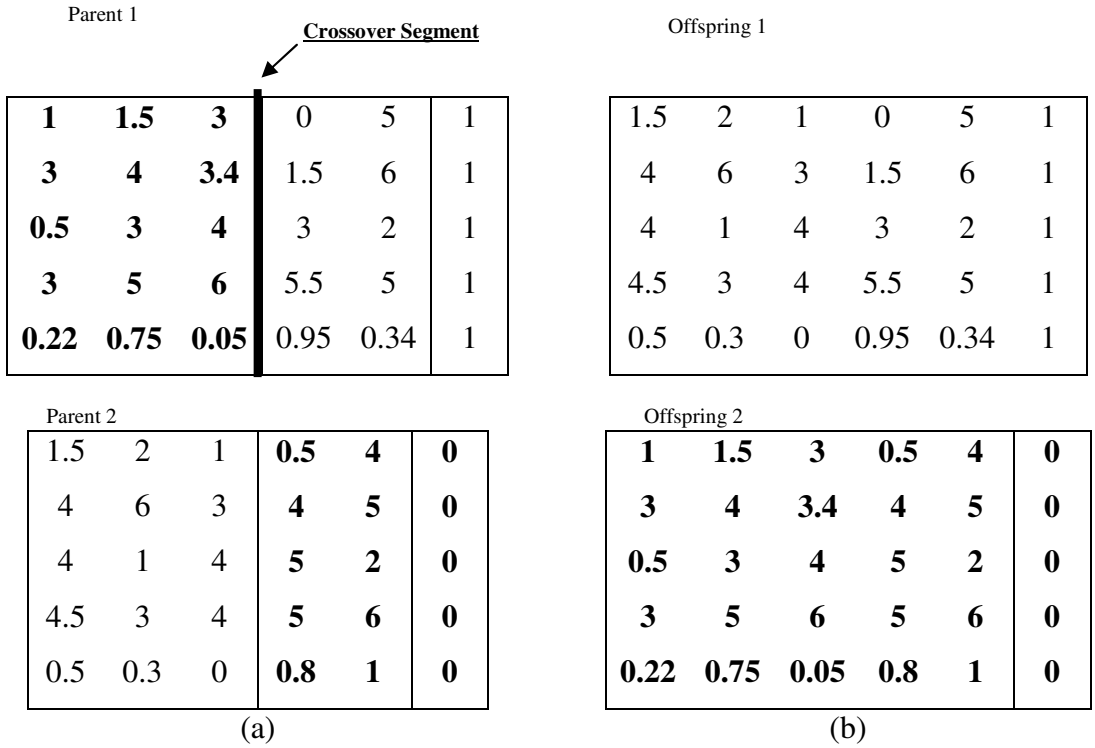


Figure 7-2. Two parents generate two offsprings through crossover; (a) parent chromosomes (b) offspring chromosomes.

7.3.1.1. Cost Estimation

At a certain step in our GA algorithm, cost (fitness) has to be estimated for a chromosome (solution). Figure **7.**3 depicts a detailed flowchart for estimating the cost using simulation when we integrate DTM with the inspection planning model. The flowchart is divided into three major parts: Virtual Inspection (VI), Dynamic Throughput Maximization (DTM) and Virtual Assembly (VA).

7.3.1.1.1. Virtual Inspection (VI)

It starts off by determining all the possible subassembly quality characteristics (*X*'s) that contribute on the final requirements (*Y*'s) and by assuming the behavior of those chosen *X*'s.

After that, enough simulations are run through CAD models (e.g. 3DCS) to find the *Y*'s that are associated with the generated *X*'s. The *Y*'s are then mapped with the *X*'s through regression. This fitted function will be needed in the final assembly part (VA) of the simulation (Note "1" in Figure **7.**3).

At the beginning, to find a cost, we generate a random variable (r^{th} subassembly item) that represents the t^{th} subassembly group according to the known probability distribution function (PDF) behavior; X_{tr} (Note "2" in Figure **7.**3). We place the generated quality characteristic into the inspection process according to a given *frequency* (Note "3" in Figure **7.**3). In order to know whether to inspect an item or not, we generate another random variable between 0 and 1 (according to a uniform distribution function that ranges from 0 to 1) and compare it to the specified frequency in the chromosome. If the random variable is less than the frequency, then we inspect the item; otherwise we do not inspect it. If the generated value (X_{tr}) was decided to be inspected and found to be in region I or IV (Figure **3.**1), then we *scrap* it (send nothing to be assembled by giving X'_{tr} a value of zero) and we add that cost of scrapping and inspecting the item. However, if it is located in regions II or III, then we *rework* it to the near nominal value and we add that cost of rework and inspection. If the part was found to be within LL_R and UL_R, then we *keep* the part as it is and we add the cost of inspection only. If the part was not inspected, then we send it directly to the batch area. Only the non-inspected, kept and reworked items are sent to the batch to be assembled. After checking all the input subassembly quality characteristic (t=1…M) for all the given items (r=1…R), we reach to a point where we have different number of subassembly items in each group because of the scrap procedure and the differences in the frequency of inspection. Suppose there are 2 subassembly groups (*a* and *b*; i.e. M=2) of size 100 (R=100) for each one of them. If 50/0 parts of subassembly groups *a*/*b* were scrapped, this means that only 50% of subassembly *b* will be utilized and 100% of the resultant *a*'s could have been utilized. At that point, we can say that the yield reduced from 100% to 50% because of the inspection. The number of parts that could be potentially assembled for this example will be 50 (Q=50 in the final inspection simulation); Q can be given as follows:

$$Q = \min\left(\sum_{r=1}^{R} step(X'_{tr}), \forall t = 1,...,M\right) \qquad (7.2)$$

Where:

X'_{tr}: Value of the quality characteristic for the t^{th} subassembly group in the r^{th} item after the inspection process

Note: $X'_{tr,i}$=0 if scrapped; $X'_{tr,i}$=nominal(X_t) if reworked

$$step(\gamma) = \begin{cases} 1, \gamma > 0 \\ 0, Otherwise \end{cases}$$

Q refers to the minimum number of non-scrapped items in all the M subassembly groups. This will reduce our yield up to this point to (R-Q).100/R%. If we have decided to implement DTM (τ_{DTM} = 1), then we go through the DTM procedure.

7.3.1.1.2. DTM Implementation

Different approaches to implement the DTM assignment can possibly be adopted. We propose a simplified approach for doing that. We only implement DTM if the number of inspected subassembly groups (η; Equation 3) equals to the number of the subassembly groups (M). This means that we will not implement it if we do not have at least one set of subassembly items that can be used to make one product. Moreover, we will not implement DTM unless we have at least two inspected items in each subassembly groups (φ, Equation 4. Refer to Note "4" in Figure **7.**3). Note that φ represents the maximum number of non-scrapped, *inspected* items among all the M subassembly groups. It can be estimated as it is show in Equation 7.4. DTM implementation is performed for φ items in each one of the M subassembly groups. Since we have to have φ number of items in each group, it is necessary to generate some virtual items with *exaggerated* values so they will not be selected in the DTM assignment. The reason for doing so is to maximize the chance for the utilizing all the inspected items in all the subassembly groups.

$$\eta = \sum_{t=1}^{M} step\left(\sum_{r} X'_{tr,i} \right) \qquad (\mathbf{7.}3)$$

$$\varphi = \max\left(\sum_{r} step\left(X'_{tr,i}\right), \forall t = 1,\ldots,M \right) \qquad (\mathbf{7.}4)$$

Where:

$X'_{tr,i}$: I*nspected* value for the r^{th} subassembly feature in the t^{th} subassembly group.

If DTM is implemented, we add the cost associated with it according to Equation 7.5.

$$C_{DTM} = c_{DTM} . \varphi . M \qquad (7.5)$$

Where:

c_{DTM}: Dynamic Throughput Maximization (DTM) cost for sorting and assembling a single final assembly.

Figure **7.**4 illustrates the procedure for sorting the batch of items for a particular simple problem instance. The output of this procedure is a matrix of sorted subassembly items to be sent to be assembled in the Virtual Assembly (VA) step. In Step 1 in Figure 7.4, we mate the φ items in each group according to a DTM algorithm, so the throughput is maximized. Let us say that the optimal throughput was found for this example to be 50% for this example (3 "good" parts out of 6). We give exaggerated values (e.g. *infinity*) for the subassembly items that were not inspected. The "good" items are sorted according to DTM and the "bad" ones are randomized in Step 2. We then enter non-inspected assembly items (U) into the matrix randomly.

The mathematical model for DTM is given in (6-10). The reason we show it here although it was presented in Chapter 5 is to present the model with the new notations in this Chapter. The objective of this model is to maximize the number of "good" produced final assemblies (φ') out of a given number of subassemblies (φ) in each subassembly groups. Notice that $\varphi' \leq \varphi$, so the throughput can be defined as φ' / φ. Constraints (7) indicate that at most one item (i) in a group (t) is chosen for a product (r). Similarly, Constraints (8) indicate that any item (i) from group (t) can be included in at most one part (r). Constraints (9) state that if a product was decided to be made, it should be fully assembled and must contain one subassembly item from each group. This constraint prevents the construction of partial or incomplete parts. Finally, Constraints (10)

indicate that the dimensional characteristic of quality characteristic *a* for each *chosen* product *r* must be bounded between all the *N* lower and upper limits (LL_a, UL_a). This mathematical model is valid for *linear* assembly functions (Constraints in 10). This formulation is different from the tolerance allocation problem. The reason is that the objective from the tolerance allocation is to specify limits in the subassembly items for the acceptance range that will possibly lead to the least failed final assemblies. Final assemblies fail if the final part is not within the tolerances specified by the customer (designer).

DTM 1

Maximize $\varphi' = \sum_{i=1}^{\varphi}\sum_{r=1}^{\varphi} z_{1ir}$ (**7.**6)

Subject to:

$$\sum_{i=1}^{\varphi} z_{tir} \le 1 \qquad \forall t = 1,...,M \ , \forall r = 1,...,\varphi \qquad (7.7)$$

$$\sum_{r=1}^{\varphi} z_{tir} \le 1 \qquad \forall t = 1,...,M \ , \forall i = 1,...,\varphi \qquad (7.8)$$

$$\sum_{i=1}^{\varphi} z_{tir} = \sum_{i=1}^{\varphi} z_{t+1,ir} \qquad \forall t = 1,...,M-1 \ , \forall r = 1,...,\varphi \qquad (7.9)$$

$$LL_a \sum_{i=1}^{\varphi} z_{1ir} \le \sum_{t=1}^{M}\sum_{i=1}^{P} x_{tir} d_{tia} \le UL_a \sum_{i=1}^{\varphi} z_{1ir} \qquad \forall a = 1,..., N \ , \forall r = 1,...,\varphi \qquad (7.10)$$

$$z_{tir} \in \{0,1\}$$

$$z_{tir} = \begin{cases} 1, \text{if item } i \text{ in group } t \text{ is used for part } r \\ 0, \text{otherwise} \end{cases}$$

Where:

LL_a, UL_a: Lower and upper limits for quality characteristic *a*

d_{tia}: Dimension of item *i* in group *t* that is associated with quality characteristics *a*

d_{tia} =0 if group *t* does not contribute in quality characteristics *a*

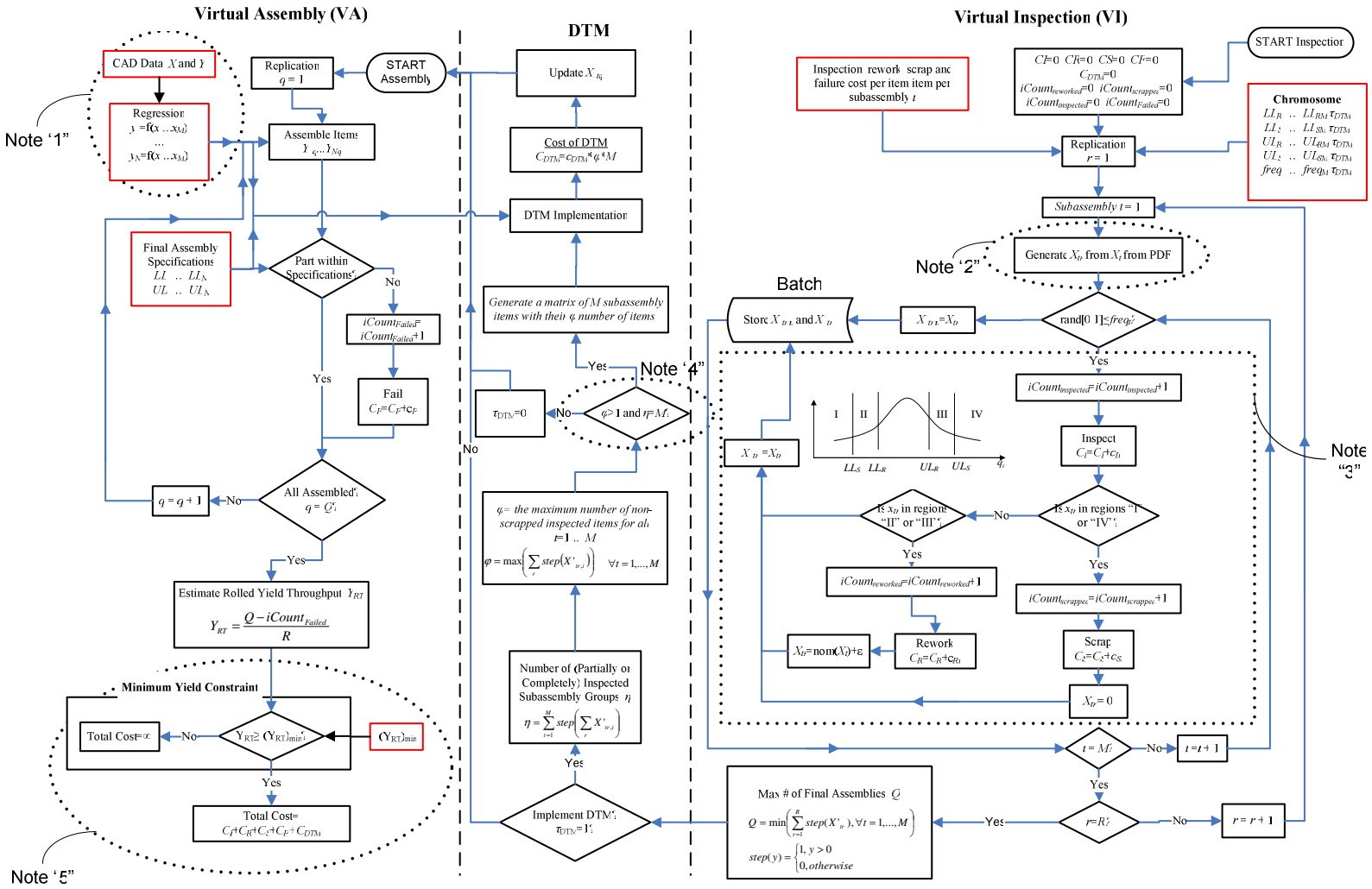


Figure 7-3. Cost estimation for inspection planning when integrating it with DTM

x_1	x_2	x_3	x_4
I	I	I	I
I	I	I	I
I	I	I	I
I	∞	I	I
∞	∞	I	I
∞	∞	I	∞

STEP 1: DTM implementation for φ (=6) items in each group

x_1	x_2	x_3	x_4
I_{DTM}	I_{DTM}	I_{DTM}	I_{DTM}
I_{DTM}	I_{DTM}	I_{DTM}	I_{DTM}
I_{DTM}	I_{DTM}	I_{DTM}	I_{DTM}
I	U	I	I
U	U	I	I
U	U	I	U

STEP 2: Hold the "good" mated final subassemblies (I_{DTM}) in their sorted locations and randomize the "bad" ones; then enter the non-inspected items randomly

Figure 7-4. DTM implementation for a Simple Problem Instance; I represents an inspected item I_{DTM} represents a sorted item according to DTM and U represents a non-inspected item

The Integer Program in equations (7.6-7.10) is computationally intensive to reach an optimal solution when the number of items in subassembly groups (φ) is large and/or when we have a large number of subassembly groups (M). Therefore, we reformulate the model by slicing the population of the subassembly quality characteristic into equal-area bins (Refer to Figure **7.**6). The modified model (DTM 2) is given in equations (7.11-7.16). Notice that if Γ=N, then DTM 2 will reduce to DTM 1.

DTM 2

Maximize $$\Gamma' = \sum_{\gamma=1}^{\Gamma}\sum_{\beta=1}^{\Gamma} z_{1\gamma\beta} \qquad (7.11)$$

Subject to:

$$\sum_{\gamma=1}^{\Gamma} z_{t\gamma\beta} \le 1 \qquad \forall t = 1,...,M\ ,\forall \beta = 1,...,\Gamma \qquad (7.12)$$

$$\sum_{\beta=1}^{\Gamma} z_{t\gamma\beta} \le 1 \qquad \forall t = 1,...,M\ ,\forall \gamma = 1,...,\Gamma \qquad (7.13)$$

$$\sum_{\gamma=1}^{\Gamma} z_{t\gamma\beta} = \sum_{\gamma=1}^{\Gamma} z_{t+1,\gamma\beta} \qquad \forall t = 1,...,M-1\ ,\forall \beta = 1,...,\Gamma \qquad (7.14)$$

$$LL_a \sum_{\gamma=1}^{\Gamma} z_{1\gamma\beta} \le \sum_{t=1}^{M} \sum_{\gamma=1}^{\Gamma} z_{t\gamma\beta} D_{t\gamma a,\min} \qquad \forall a = 1,\ldots,N\ ,\forall \beta = 1,\ldots,\Gamma \qquad (\mathbf{7.}15)$$

$$UL_a \sum_{\gamma=1}^{\Gamma} z_{1\gamma\beta} \ge \sum_{t=1}^{M} \sum_{\gamma=1}^{\Gamma} z_{t\gamma\beta} D_{t\gamma a,\max} \qquad \forall a = 1,\ldots,N\ ,\forall \beta = 1,\ldots,\Gamma \qquad (\mathbf{7.}16)$$

$$z_{t\gamma\beta} \in \{0,1\}$$

Where:

$$z_{t\gamma\beta} = \begin{cases} 1, \textit{ if bin } \gamma \textit{ in subassembly t is used to make sub} - \textit{batch } \beta \\ 0, \textit{ otherwise} \end{cases}$$

$D_{t\gamma a,min}$: the lower limit dimension of the γ^{th} bin in the t^{th} subassembly for quality characteristic a (Figure **7.**6)

$D_{t\gamma a,max}$: the upper limit dimension of the γ^{th} bin in the t^{th} subassembly for quality characteristic a (Figure **7.**6)

Γ: Number of sub-batches (bins)

γ: Bin number of a subassembly group

β: Bin (sub-batch) number in a product (final assembly)

Although solving (7.11-7.16) could be a good way to simplify the problem, the size of achieved throughput is governed by the size of the sub-batches (bins). Moreover, DTM 1 and 2 do not consider when we have nonlinear assembly functions. Therefore, we have resorted to solve the problem using a simplified approach that was presented in Chapter 6 with Simulated Annealing (SA) algorithm that is explained in Section 3.1.2.

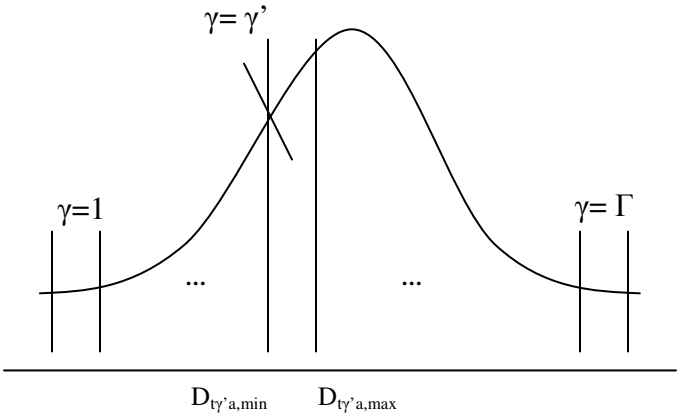


Figure 7-5. DTM 2 representation for a sliced subassembly group

7.3.1.1.3. Virtual Assembly (VA)

We take a set of input quality characteristics (*X*'s) and find the associated final quality character tics (*Y*'s) according to the function found after fitting the CAD data. If the part (*Y*) is found to be within specifications (*LL* and *UL*), then we proceed with the next step. However, if it does not fall within the specification limits, then we fail it and we add the failure cost. After examining all the *Q* parts, we estimate the yield based on that. We notice that the final yield is dependent on the scrap and failure rates, as follows:

$$Y_{RT} = \frac{Q - Q_F}{R}$$

Where:

Y_{RT} : Rolled Yield Throughput

Q : Maximum number of items in a subassembly to be assembled (Equation 7.2)

R : Number of subassembly parts before inspection

Q_F : Number of failed subassembly parts after the final assembly, $Q_F = iCount_{Failed}$ (Figure **7.**3)

If the yield was found to be less than the pre-specified minimum yield, then we return a total cost of *infinity*, so that particular inspection plan will be eliminated (Note "5" in Figure **7.**3).

7.3.1.2. Simulated Annealing Algorithm (SADTM)

The solution (S) for M subassembly groups with φ items in each group can be represented as shown in Equation 7.17. The size of the sequence is $\varphi.M$. The first M items in S represent part 1, the next M items represent part 2, and so on.

$$S = \left\{ \underbrace{s_1\ s_2\ s_3\ \ldots s_M}_{Part\ 1};\ \underbrace{s_{M+1}\ s_{M+2}\ s_{M+3}\ \ldots s_{2M}}_{Part\ 2};\ \ldots;\ \underbrace{s_{(\varphi-1)M+1}\ \ldots s_{\varphi M}}_{Part\ \varphi} \right\} \qquad (7.17)$$

The algorithm is performed as follows: after generating initial solution, we estimate the throughput achieved. Then, we find a neighboring solution by modifying the solution of β of the φ parts in S. We accept the new generated solution if it is better than the previous one. Also, if the generated solution is worse than the previous solution, we accept the generated solution with a probability that reduces according to the annealing schedule as the algorithm proceeds. We assume that s_1=1, s_{M+1}=2,…, $s_{(\varphi-1)M+1}$=φ

For a given annealing schedule: $Temperature = \{t_1, t_2, ...\}$, the algorithm goes as follows:

Step 1 (a) Set *Epoch* to 1 (*Epoch* = 1).

(b) Generate initial sequence randomly

Step 2 Estimate throughput for the initial solution S_1: $T(S_1)$

Step 3 Generate a temporary solution; S^*:

(l) Set i = 2 (second item in S), r = 1 (first part)

(m)Generate a random variable c from U(0,1)

(n) If $c \leq \beta$, generate new subassembly items for part r: $S'_i = \{s'_i\ s'_{i+1} \ldots s'_{i+M-2}\}$;

otherwise $S'_i = \{s_i\ s_{i+1},\ \ldots,\ s_{i+M-2}\}$

(o) If $r > 1$ (not the first part), continue; otherwise skip to *Step* 3(i)

(p) $j = i$

(q) If $s'_j \in \{..., s'_{j-2M}, s'_{j-M}\}$, generate new $s'_j \notin \{..., s'_{j-2M}, s'_{j-M}\}$; otherwise continue

(r) $j = j + 1$

(s) If $j \leq i+M-2$, go to *Step* 3(f); otherwise continue

(t) $i = i + M$, $r = r + 1$

(u) If $i \leq \varphi.M$, go to *Step* 3(b); otherwise continue

(v) $S^* = \{1\ S'_1\} \cup \{2\ S'_2\} \cup ... \cup \{\varphi\ S'_{\varphi.M}\}$

Step 4 Estimate throughput for the temporary sequence S^*: T(S^*)

Step 5 If $\Delta T = T^* - T_{Epoch} \leq 0$ (non-improving solution), go to *Step* 7; otherwise (improving solution) go to *Step* 9

Step 6 Generate a random variable *b* from U(0,1)

Step 7 If $b \geq e^{-\Delta T / t_{Epoch}}$ (Do not accept non-improving solution): *Epoch* = *Epoch* +1, *S*(*Epoch*)=*S*(*Epoch*-1), *T*(*Epoch*)=*T*(*Epoch*-1), then go to Step 3; otherwise continue

Step 8 (a) *Epoch*=*Epoch*+1

(b) $S_{Epoch}=S^*$ and $T_{Epoch}=T^*$

Step 9 If a stopping criterion is met, stop; otherwise go to *Step* 3

7.4. NUMERICAL EXAMPLE

In this Section, we solve a numerical example by integrating the inspection planning with DTM. We want to develop an inspection plan for a product that has four input quality characteristics (*X*'s, *M*=4) and three final quality characteristics (*Y*'s, *N*=3). The behavior of *Y*'s when *X*'s change can be mapped after simulating the product using 3DCS or VSA. After that, we can relate the *X*'s with the *Y*'s by using regression. The objective is to determine the optimal frequencies, binary DTM decision variable and the action limits (rework and scrap limits) that give the minimum total cost. We also consider that we are constrained with a minimum rolled yield throughput. Lower Limits for the 3 final quality characteristics are respectively: 80, 35 and 3.5

and the Upper Limits are respectively: 120, 42, and 5. The mapping functions between the x's and the y's are as follows:

$$y_1 = 100 - 50x_1 + x_2 - x_3 + 50x_4,$$

$$y_2 = 35 - x_1 + x_2 - x_3 + 12x_4,$$

$$y_3 = 2.5 - x_1 + x_2 + 2x_4.$$

Given: R=50 (Initial number of items in a subassembly; data shown in Table **7.**1) , Minimum Yield = 45%, Number of GA Replications = 1000, c_I = \$0 (cost of inspecting a single quality characteristic); c_R = \$1080 (cost of reworking a single quality characteristic); c_S = \$925 (cost of scrapping a single quality characteristic); c_F = \$300 (cost of failing a single final assembly), c_{DTM}=\$0 (cost of implementing DTM assembly on a single item in subassembly group t); β = 0.1 (SADTM parameter); Population Size: 6 Chromosomes, Mutation Rate: 20%. We execute our SADTM for 30 seconds if DTM is to be implemented. All the computations were executed in MATLAB using a Pentium 4 processor, 3.0 GHz and 3.0 GB RAM.

Table 7.1. Subassembly quality characteristic values

No.	x_1	x_2	x_3	x_4	No.	x_1	x_2	x_3	x_4
1	0.2691	0.4342	0.2036	0.8679	**26**	0.1446	0.3928	0.1544	0.1767
2	0.6741	0.7312	0.014	0.4048	**27**	0.7411	0.0665	0.7354	0.1121
3	0.3895	0.2461	0.9575	0.4564	**28**	0.4711	0.1618	0.2962	0.5329
4	0.9243	0.7756	0.0731	0.6847	**29**	0.6426	0.4909	0.3211	0.9805
5	0.8864	0.6033	0.3856	0.5511	**30**	0.6503	0.1093	0.2825	0.0336
6	0.0041	0.5659	0.2038	0.6198	**31**	0.3393	0.3814	0.4709	0.4656
7	0.8022	0.3343	0.2208	0.9983	**32**	0.3856	0.7196	0.3296	0.6511
8	0.6298	0.4679	0.6406	0.1716	**33**	0.3147	0.8526	0.7887	0.8371
9	0.0542	0.2835	0.0416	0.0335	**34**	0.0387	0.9653	0.5223	0.0769
10	0.8294	0.5543	0.2356	0.8899	**35**	0.5322	0.377	0.0227	0.466
11	0.5217	0.5746	0.26	0.8891	**36**	0.0369	0.8077	0.4131	0.4013
12	0.1522	0.3721	0.8637	0.7708	**37**	0.5186	0.9028	0.2573	0.8079
13	0.5694	0.3011	0.1924	0.5906	**38**	0.6562	0.65	0.9361	0.3077
14	0.2574	0.9742	0.6693	0.1485	**39**	0.2745	0.8272	0.7008	0.7512

15	0.4167	0.1512	0.3815	0.5905	**40**	0.5609	0.6854	0.3529	0.5792
16	0.9075	0.464	0.9862	0.8477	**41**	0.7655	0.6623	0.9601	0.8407
17	0.7609	0.2229	0.7732	0.3186	**42**	0.2471	0.9866	0.0637	0.2876
18	0.0902	0.345	0.2307	0.4079	**43**	0.4887	0.0193	0.8599	0.505
19	0.1907	0.0523	0.4983	0.2045	**44**	0.9642	0.568	0.8147	0.1445
20	0.5599	0.8694	0.7243	0.6216	**45**	0.3378	0.3032	0.4925	0.0779
21	0.3595	0.3207	0.9972	0.3653	**46**	0.7353	0.7508	0.5	0.8766
22	0.0514	0.4154	0.6779	0.5715	**47**	0.9196	0.5355	0.4584	0.685
23	0.9621	0.8327	0.8897	0.277	**48**	0.9535	0.2246	0.9628	0.9595
24	0.0749	0.5142	0.5298	0.3414	**49**	0.6774	0.3445	0.6932	0.2275
25	0.4551	0.1843	0.7002	0.2885	**50**	0.1073	0.1021	0.5727	0.9691

The problem instance was solved when we implement the integration and the results are shown in Table **7.**2. We notice that the found solution suggests that we almost do no rework at all. It is either we scrap an item (which is not so likely) or to implement DTM procedure. We solved the same problem with *arbitrary* assembly. The solution is shown in Table **7.**3. Notice that when we integrate DTM with our inspection planning, we reduced the cost dramatically by more than 4.5 times. The execution time for the hybrid implementation is considerably long (53 hours). This is because we set SADTM implementation for 30 seconds every time we estimate the cost. Since inspection plans are usually strategic that could result in substantial cost savings, running a code for that long time could be justifiable for certain cases.

Table 7.2. Solution when assembly is based on DTM; execution time for 1000 replications: 53.2 hours

Decision Variables	X_1	X_2	X_3	X_4	Total Cost; $
*LLR**	5	4	3	4	4,800
*LLS**	5	5	3.5	4.5	
*ULR**	2	5	4	2.5	
*ULS**	3	5	4	5	
*freq**	0.90	0.80	0.70	1.00	

Table 7.3. Solution when assembly is arbitrary; execution time for 1,000,000 replications: 26.3 hours

Decision Variables	X_1	X_2	X_3	X_4	Total Cost; $
*LLR**	2	4.5	3	0.5	21,830
*LLS**	3.5	4.5	3.5	4	
*ULR**	1.5	5	4.5	4	
*ULS**	1.5	5	5	4.5	
*freq**	0.80	0.10	0.8	0.6	

7.5. SUMMARY

We can summarize our findings and conclusions as follows:

1.We presented an approach in this work for reducing the inspection planning cost by integrating the inspection plan with subassembly mating in a dynamic-basis. As we expected, we verified by a numerical example that the cost of inspection can be substantially decreased when implementing the proposed hybrid variation strategy.

2.Generating inspection plans can be considered as a strategic plan, therefore the long execution time for a problem instance of practical size would be often justifiable.

3.The accuracy of the solution depends on several parameters to be chosen. One of those parameters is the *execution time* for the Simulated Annealing algorithm to solve the dynamic yield maximization. The longer the execution of SA, the more likely more possible savings would be discovered.

4.The Dynamic Rolled Yield Throughput Maximization procedure would be more beneficial when the failure cost of the final assembly is high. The failure cost can be minimized by implementing rework/scrap procedure, DTM procedure or combination of both.

5.We proposed different approaches to solve the DTM problem in Chapter 7. We found that using Ant Colony Motivation (ACO) algorithm outperforms the used Simulated Annealing (SA) Algorithm in terms of finding higher throughputs in less time. However, implementing it was found to be more complicated than SA. We are considering using the proposed ACO approach in order to decrease the computation time as much as possible.

6.We are looking forward to implementing the proposed approach to a real-life problem where potential savings could be discovered.

CHAPTER 8: SUMMARY AND FUTURE RESEARCH

The objective of this dissertation was to present new strategic and dynamic variation reduction techniques in assembly. In this Chapter, we summarize what we proposed, implemented, achieved, and what we propose for future research.

The Strategic Variation Reduction model based on an inspection plan was proposed in Chapter 3. It was based on Monte Carlo Simulation, and a Genetic Algorithm (GA). New decision variables were considered in this problem. These decision variables (frequency of inspection, 4 limits that determine whether to rework, scrap or keep an assembly item) have not been presented in the literature before. It was computationally validated that using these decision variables can lead to considerable savings when the yield is a known constraint. This is true especially when the cost of failure is generally less than the inspection, rework and scrap costs. Referring to Example 1 in Chapter 3, we found that by refining the resolution of frequency from {0, 1} to {0, 0.1, 0.2, … , 0.9, 1}, we can potentially save about 50% of the original cost. This suggests that frequency is a necessary decision variable to be considered. In addition to existing products, the model was extended to develop inspection plans for newly launched products where historical data are not available. Simulated data generated by CAD simulation software such as DCS and VSA is proposed to be used. This is a considerable contribution in the area of early stage design. We use this simulated data to map the inputs (tolerances) to the outputs (measures) through the CAD software (e.g. 3DCS). We would recommend providing such data (X's vs. Y's) more directly as an output file when running a simulation using the CAD software. Alternatively, we can integrate the proposed algorithm with the CAD software. We used linear regression to fit (map) the X's with Y's (CAD data). A test for quality of fit is may be desirable to make sure that our model is statistically valid. On the other hand, we can run a variable selection procedure that can optimally select the most contributing variables. One possible approach is to implement stepwise regression. Although it was assumed constant, the cost parameters in the inspection cost function are usually different from one subassembly group to another. The cost function was simplified to obtain intuitive conclusions. We believe it is not difficult to generalize this in our simulation, to make the proposed model more applicable.

The second part of this dissertation is Dynamic Variation Reduction. We introduced Dynamic Variation Reduction (DVR) and Dynamic Rolled Yield Throughput Maximization models (DTM). Simple rules were proposed for simple cases for the DVR model. Those rules were proven to be optimal for simple cases (2 subassembly groups and 1 quality characteristics). For more complicated cases, we provided additional rules to solve for subassembly with more than 2 subassembly groups and 1 quality characteristics. We solved the Dynamic Rolled Yield Throughput (DTM) using different heuristics, such as greedy heuristics, and Metaheuristics (Simulated Annealing and Ant Colony Optimization.) In this work, we assumed that all the products are made of a single level of assembly. This means that we only studied simple products. Relaxing such assumption could be very interesting to be further studied because most of the products are made of multi-levels of subassemblies (e.g. engines, pumps, etc.)

We are aware that the performances of the Genetic Algorithm, Ant Colony, and Simulated Annealing solutions depend highly on the used parameters. Therefore, it would be interesting to conduct statistical analyses to determine the best parameters.

The third part of this dissertation involved a hybrid implementation of the strategic and dynamic variation reduction techniques. We performed computational experiments, confirming that such implementation will lead to very significant savings.

Our work here was based on simulation that can potentially be validated if it is implemented in a real manufacturing facility. We believe implementing the methods we have proposed can lead to great deal of improvement in quality, and reduction of total cost.

BIBLIOGRAPHY

Agapiou, J. S.; Steinhilper, E.; Gu, F. and Bandyopadhyay, P., (2003), "A Predictive Modeling Methodology for Part Quality from Machining Lines," Transactions of the North American Manufacturing Research Institution/SME, Vol. 32, pp. 629-636.

Ant Colony Optimization, Dorigo, M. and Stützle, T., The MIT Press, 2004.

Azam, M., Pattipati, K. R. and Patterson-Hine, A., (2004), "Optimal Sensor Allocation for Fault Detection and Isolation", IEEE International Conference on Systems, Man and Cybernetics, The Hague, Netherlands.

Azam, M., Pattipati, K. R. and Patterson-Hine, A., 2004, "Optimal Sensor Allocation for Fault Detection and Isolation", IEEE International Conference on Systems, Man and Cybernetics, Vol. 2, pp. 1309-1314.

Binglin, Z., Tinghu, Y. and Ren, H., (1993), "A Genetic Algorithm for Diagnosis Problem Solving", in Proc. Int. Conf. SMC, vol.2 , pp. 404-408.

Blum, C. and Roli, A., (2003) "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison," ACM Computing Surveys, Vol. 35, pp. 268-308.

Boyer, D. E. and Nazemetz (1985), "Introducing Statistical Selective Assembly – A Means of Producing High Precision Assemblies from Low Precision Components," Annual International Industrial Engineering Conference Proceedings.

Byrne, D. M. and Taguchi, S., 1987¸ "The Taguchi Approach to Parameter Design," Quality Progress, 20, pp. 19-26.

Ceglarek, D.; Huang, W.; Zhou, S.; Ding, Y; Kumar, R. and Zhou Y., (2004), "Time-based Competition in Manufacturing: Stream-of-Variation Analysis (SOVA) Methodology-Review," International Journal of Flexible Manufacturing Systems, 16/1, pp.45-64.

Chan, W. M. and Ibrahim, R. N., (2004), "Evaluating the quality level of a product with multiple quality characteristics," International Journal of Advanced Manufacturing Technology, 24, Numbers 9-10, pp. 738-742.

Chase, K. W., Greenwood, W. H., Loosli, B. G and Hauglund, L. F., (1990), "Least Cost Tolerance Allocation for Mechanical Assemblies with Automated Process Selection," Manufacturing Review, 3/1, pp. 49-59.

Chen, S.-L. and Chung, K.-J., (1996), "Selection of the Optimal Precision Level and Target Value for a Production Process: The Lower-Specification-Limit Case," IIE Transactions, 28, pp. 979-985.

Chen, S.-L. and Chung, K.-J., 1996, "Selection of the Optimal Precision Level and Target Value for a Production Process: The Lower-Specification-Limit Case," IIE Transactions, 28, pp. 979-985.

Chen, T. J. and Thornton, A. C. 1999, "Quantitative Selection of Inspection Plan," Design Lemma and Methodology Conference, ASME Design Technical Conferences, Las Vegas, NV.

Chen, T. J. and Thornton, A. C., (1999), "Quantitative Selection of Inspection Plan," Design Remark and Methodology Conference, ASME Design Technical Conferences, Las Vegas, NV.

Desmond, D. J. and Setty, C. A., (1962) "Simplification of Selective Assembly," International Journal of Production Research, Vol. 1, pp. 3-18.

Dimensional Control System (3DCS), Tutorial Manual, 2005.

Ding, Y., Kim, P., Ceglarek, D. and Jin, J., (2003), "Optimal Sensor Distribution for Variation Diagnosis in Multi-station Assembly Processes," IEEE Transactions on Robotics and Automation, Vol. 19(4), pp. 543-556.

Ding, Y., Kim, P., Ceglarek, D. and Jin, J., "Optimal Sensor Distribution for Variation Diagnosis in Multi-station Assembly Processes," IEEE Transactions on Robotics and Automation, Vol. 19(4), pp. 543-556, 2003.

Dorigo M., V. Maniezzo & A. Colorni (1996), "Ant System: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics-Part B,* 26(1):29-41.

Frey, D. D.; Otto, K. N.; and Wysocki, J., (2000), ''Evaluating Process Capability during the Design of Manufacturing Systems,'' Journal of Manufacturing Science and Engineering, 122, pp. 513–519.

Gao, J., Chase, K. W. and Magleby, S. P., (1995), "Comparison of Assembly Tolerance Analysis by the Direct Linearization and Modified Monte Carlo Simulation Methods," Proceedings of the ASME Design Engineering Technical Conferences, Boston, MA, pp. 353-360.

Gen, M. and Cheng, R., Genetic Algorithms and Engineering Design, Wiley and Sons, 1997.

GE Annual Report of the year 1998, url: http://www.ge.com/annual98/share/index.htm. Last time accessed: 09/24/2006.

Greenstein, E. and Rabinowitz, G., 1997, "Double-Stage Inspection for Screening Multi-Characteristic Items," IIE Transactions, 29, pp. 1057-1061.

Harry, M. J. and Lawson, J. R., Six Sigma Producibility Analysis and Process Characterization, Addison Wesley, Reading, MA, 1992.

Hu, S.J., (1997), "Stream of Variation Theory for Automotive Body Assembly", *Annals of CIRP*, Vol.46/1, pp.1-6.

Introduction to Robust Design (Taguchi Method), iSix Sigma Website, Accessed on: March 13th, 2005, URL: http://www.isixsigma.com/

Kalos and Whitlock, Monte Carlo Methods, Vol. 1, Wiley Sons, 1986.

Kannan, S. M.; Jayabalan, V. and Jeevanantham, K., 2003, "Genetic Algorithm for Minimizing Assembly Variation in Selective Assembly," International Journal of Production Research, Vol. 41, No. 14, pp. 3301 – 3313.

Kannan, SM and Jayabalan, V. (2001), "A New Grouping Method for Minimizing the Surplus Parts in Selective Assembly," Quality Engineering, Vol. 14, 1, pp. 67-75.

Kannan, SM.; Jayabalan, V. and Jeevanantham, K., (2003) "Genetic Algorithm for Minimizing Assembly Variation in Selective Assembly," International Journal of Production Research, Vol. 41, No. 14, pp. 3301 – 3313.

Khan, A., Ceglarek, D., Ni, J., (1998) "Sensor Location Optimization for Fault Diagnosis in Multi-Fixture Assembly Systems," Trans. of ASME, Journal of Manufacturing Science and Engineering, Vol. 120, No. 4, pp. 781-792, also in 1996 ASME International Mechanical Engineering Congress and Exposition, MED-4, pp. 473-484, Atlanta, GA.

Li, M.-H. C., (2002), "Optimal Process Setting for Unbalanced Tolerance Design with Linear Loss Function," Journal of the Chinese Institute of Industrial Engineers, 19/5, pp. 17-22.

Lindsay, G. F. and Bishop, A. B., (1964) "Allocation of screening inspection effort—a dynamic programming approach," Manage. Sci., vol. 10, pp. 342–352.

Agrawala, M., Phan, D., Heiser, J., Haymaker, J., Klingner, J. Hanrahan, P., Tversky, B. (2003), "Designing Effective Step-By-Step Assembly Instructions," International Conference on Computer Graphics and Interactive Techniques, pp.828-837.

Mandroli, S. S.; Shrivastava, A. K. and Ding, Y., 2006, "A Survey of Inspection Strategy and Sensor Distribution Studies in Discrete-Part Manufacturing Processes," IIE Transactions, Vol. 38, pp. 309-328.

Mansoor, E. M. (1961), "Selective Assembly – Its Analysis and Applications," International Journal of Production Research, Vol. 1, pp. 13-24.

Mease, D.; Nair, V. N. and Sudjianto, A., (2004), "Selective Assembly in Manufacturing: Statistical Issues and Optimal Binning Strategies," Technometrics, Vol. 46, 2, pp. 165-175.

Mechanical Assemblies, Daniel E. Whitney, 2004, Oxford University Press, Inc.

Meller, R. D. and Bozer, Y. A., (1996) "A New Simulated Annealing Algorithm for the Facility Layout Problem," International Journal of Production Research, Vol. 34, No. 6, pp. 1675 – 1692.

Mitsuo Gen and Runwei Cheng, Genetic Algorithms and Engineering Design, 1997, John Wiley & Sons.

Musa, R. and Chen, F. F., 2006, "Simulated Annealing and Ant Colony Optimization Algorithms for the Dynamic Rolled Yield Throughput Maximization Problem," Accepted for publication in the International Journal of Advanced Manufacturing Technology.

Musa, R.; Chen, F. F. and Ghoniem, A. S., (2006a) "Dynamic Variation Reduction Technique in Assembly Lines after Batch Inspection," Industrial Engineering Research Conference (IERC) Proceedings.

Musa, R.; Chen, F. F. and Sturges, R. H., (2006b) "Simple Rules to Achieve Variation Reduction Based on Optimal Subassembly Part Mating," International Symposium on Flexible Manufacturing (ISFA), Osaka, Japan.

Musa, R.; Sturges, R. H. and Chen, F. F., (2006c) "Agile Inspection Planning Using CAD Data," Computer-Aided Design and Applications, Vol. 3, Nos. 1-4, pp. 69 – 78.

Musa, R.; Chen, F. F. (2006) "Hybrid Implementation of Agile Inspection Planning and Dynamic Throughput Maximization," Sent to a journal for publication.

Pignatiello, J. J., "Strategies for Robust Mutiresponse Quality Engineering," IIE transactions, 25/3, pp. 5-15, 1993.

Peng, Y and Reggia, J., "A Connectionist Model for Diagnostic Problem Solving," IEEE Transactions on Systems, MAN, and Cybernetics, Vol. 19, No. 2, 1989, pp. 285-298.

Pugh, G. A. (1986), "Partitioning for Selective Assembly," Computers and Industrial Engineering, Vol. 11, pp. 175-179.

Pugh, G. A. (1992), "Selective Assembly with Components of Dissimilar Variance," Computers and Industrial Engineering, Vol. 23, 1-4, pp. 487-491.

Ross, P., Taguchi Techniques for Quality Engineering, McGraw Hill, 1988.

Shakeri, M., Pattipati, K., Raghavan, V. and Patterson-Hine, A. , (1998), "Optimal and Near Optimal Algorithms for Multiple Fault Diagnosis with Unreliable Tests", IEEE Transactions on SMC, part C, vol. 28, no. 3, pp. 431-440.

Taguchi, G., Introduction to Quality Engineering, Asian Productivity Organization, 1988.

Tongco, E. and Meldrum, D., (1996), "Optimal Sensor Placement of Large Flexible Space Structures," Journal of Guidance, Control and Dynamics, Vol. 19, No. 4, pp. 961-963.

Tovey, C. A., (2002) "Tutorial on Computational Complexity," Interfaces, Vol. 32, No. 3, pp. 30 – 61.

Tsui, K.-L., (1999), "Robust Design Optimization for Multiple Characteristic Problems," International Journal of Production Research, 37/2, pp. 433-445.

Wade, O. R., Tolerance Control, Chapter 2 in Tool and Manufacturing Engineers Handbook, Vol. 1, Machining, Drozda, T. J and Wicks, C. (editors), Society of Manufacturing Engineers (SME), Dearborn, MI.

Whitney, D. E., 2004, Mechanical Assemblies, Oxford University Press, Inc.

Yin, P.-Y., and Wang, J.-Y., (2006) "Ant Colony Optimization for the Nonlinear Resource Allocation Problem," Applied Mathematics and Computations, Vol. 74, pp. 1438 – 1453.

Zhong, W., (2004), "Unified Modeling of Variation Propagation and Tolerance Synthesis for Integrated Machine-Assembly Systems, Part II: System Level Tolerance Synthesis," Transactions of the North American Manufacturing Research Institution/SME, Vol. 32, pp. 549-556.

Zhong, W., Hu, S. J., Bingham, D., (2002), "Selecting Process Parameters and Machine Tolerances for Optimal System Performance," International Conference on Frontiers of Design and Manufacturing, 5th S. M. Wu Symposium on Manufacturing Science, ASME, Dalian.

APPENDIX

Proof for Remark 4.1 (a)

We prove by contradiction that implementing the proposed rule results in optimally minimizing the objective function; $F = F'+F''$ (Refer to Equations A1 and A2). We prove that for both the first (F') and the second terms (F'') in the objective function separately.

$$F' = \sum_{p=1}^{P}\left(\left|nom - \sum_{g=1}^{2}\sum_{i=1}^{P} x_{gip} d_{gi}\right|\right) \qquad \text{(A1)}$$

$$F'' = \sum_{p=1}^{P}\left(\left|\sum_{g=1}^{2}\sum_{i=1}^{P} x_{gip} d_{giq} - (1/P)\sum_{p=1}^{P}\sum_{g=1}^{2}\sum_{i=1}^{P} x_{gip} d_{gi}\right|\right) \qquad \text{(A2)}$$

First: Proof for the first term; *F*'.

Solution 1: We sort the first group (A) from minimum to maximum and the second group (B) from maximum to minimum as follows. Then, we mate (assign) a_1 with b_p, a_2 with $b_{p\text{-}1}$, etc. Notice that this solution is feasible.

Group *A*: $a_1 \le a_2 \le ... \le a_i \le ... \le a_j \le ... \le a_{P-1} \le a_P$

Group *B*: $b_P \ge b_{P-1} \ge ... \ge b_{P+1-i} \ge ... \ge b_{P+1-j} \ge ... \ge b_2 \ge b_1$

The following is always valid, $a_j \ge a_i$ and $b_{P+1-i} \ge b_{P+1-j}$, therefore the following is true:

$$a_j + b_{P+1-i} \ge \left\{\begin{array}{c} a_i + b_{P+1-i} \ge a_j + b_{P+1-j} \\ or \\ a_j + b_{P+1-j} \ge a_i + b_{P+1-i} \end{array}\right\} \ge a_i + b_{P+1-j} \qquad \text{(A3)}$$

The objective function for solution 1 can be expressed as follows:

$$F'_1 = \alpha + \beta_1 + \gamma + \delta_1 + \lambda \qquad \text{(A4)}$$

Where:

$$\alpha = \left|nom - (a_1 + b_P)\right| + \left|nom - (a_2 + b_{P-1})\right| + ... + \left|nom - (a_{i-1} + b_{P+2-i})\right| \qquad \text{(A5)}$$

$$\beta_1 = \left|nom - (a_i + b_{P+1-i})\right| \qquad \text{(A6)}$$

$$\gamma = \left|nom - (a_{i+1} + b_{P-i})\right| + \left|nom - (a_{i+2} + b_{P-1-i})\right| + ... + \left|nom - (a_{j-1} + b_{P+2-j})\right| \qquad \text{(A7)}$$

$$\delta_1 = \left|nom - (a_j + b_{P+1-j})\right| \qquad \text{(A8)}$$

$$\lambda = \left|nom - (a_{j+1} + b_{P-j})\right| + \left|nom - (a_{j+2} + b_{P-1-j})\right| + ... + \left|nom - (a_P + b_1)\right| \qquad \text{(A9)}$$

Suppose that the following solution (solution 2) improves the solution provided in solution 1, i.e. $F'_2 \le F'_1$. In solution 2, notice that we only swapped the sequence of two items (b_{P+1-j}, b_{P+1-i}) in group *b*.

Solution 2:

Group *A*: $a_1, a_2, ..., a_i, ..., a_j, ..., a_P$

Group *B*: $b_P, b_{P-1}, ..., \underline{b_{P+1-j}}, ..., \underline{b_{P+1-i}}, ..., b_1$

The objective function for solution 2 can be expressed as follows:

$$F'_2 = \alpha + \beta_2 + \gamma + \delta_2 + \lambda \qquad \text{(A10)}$$

Where:

$$\beta_2 = \left|nom - (a_i + b_{P+1-j})\right| \qquad \text{(A11)}$$

$$\delta_2 = \left|nom - (a_j + b_{P+1-i})\right| \qquad \text{(A12)}$$

And α, γ, λ are the same as in A5, A7 and A9.

Referring to A3, there are six cases to compare F'_1 with F'_2:

1. $a_j + b_{P+1-i} \le nom$

$$\beta_1 + \delta_1 = 2.nom - (a_i + b_{P+1-i} + a_j + b_{P+1-j})$$

$$\beta_2 + \delta_2 = 2.nom - (a_i + b_{P+1-j} + a_j + b_{P+1-i})$$

$$\beta_1 + \delta_1 = \beta_2 + \delta_2$$

Therefore, $F'_1 = F'_2$

2. $\max(a_i + b_{P+1-i}, a_j + b_{P+1-j}) \le nom < a_j + b_{P+1-i}$

$$\beta_1 + \delta_1 = 2.nom - (a_i + b_{P+1-i} + a_j + b_{P+1-j})$$

$$\beta_2 + \delta_2 = nom - a_i - b_{P+1-j} + a_j + b_{P+1-i} - nom = (a_j - a_i) + (b_{P+1-i} - b_{P+1-j})$$

$$(\beta_1 + \delta_1) - (\beta_2 + \delta_2) = 2(nom - (a_j + b_{P+1-i})) < 0$$

Therefore, $F'_1 < F'_2$

3. $a_j + b_{P+1-j} \le nom < a_i + b_{P+1-i}$

$$\beta_1 + \delta_1 = a_i + b_{P+1-i} - nom + nom - a_j - b_{P+1-j} = (a_i - a_j) + (b_{P+1-i} - b_{P+1-j})$$

$$\beta_2 + \delta_2 = nom - a_i - b_{P+1-j} + a_j + b_{P+1-i} - nom = (a_j - a_i) + (b_{P+1-i} - b_{P+1-j})$$

$$(\beta_1 + \delta_1) - (\beta_2 + \delta_2) = 2(a_i - a_j) \le 0$$

Therefore, $F'_1 \le F'_2$

4. $a_i + b_{P+1-i} \le nom < a_j + b_{P+1-j}$

$$\beta_1 + \delta_1 = nom - a_i - b_{P+1-i} + a_j + b_{P+1-j} - nom = (a_j - a_i) + (b_{P+1-j} - b_{P+1-i})$$
$$\beta_2 + \delta_2 = nom - a_i - b_{P+1-j} + a_j + b_{P+1-i} - nom = (a_j - a_i) + (b_{P+1-i} - b_{P+1-j})$$

$$(\beta_1 + \delta_1) - (\beta_2 + \delta_2) = 2(b_{P+1-j} - b_{P+1-i}) \le 0$$

Therefore, $F'_1 \le F'_2$

5. $a_i + b_{P+1-j} \le nom < \min(a_i + b_{P+1-i}, a_j + b_{P+1-j})$

$$\beta_1 + \delta_1 = a_i + b_{P+1-i} + a_j + b_{P+1-j} - 2.nom$$
$$\beta_2 + \delta_2 = nom - a_i - b_{P+1-j} + a_j + b_{P+1-i} - nom = (a_j - a_i) + (b_{P+1-i} - b_{P+1-j})$$

$$(\beta_1 + \delta_1) - (\beta_2 + \delta_2) = 2((a_i + b_{P+1-j}) - nom) \le 0$$

Therefore, $F'_1 \le F'_2$

6. $nom < a_i + b_{P+1-j}$

$$\beta_1 + \delta_1 = (a_i + b_{P+1-i} + a_j + b_{P+1-j}) - 2.nom$$
$$\beta_2 + \delta_2 = (a_i + b_{P+1-j} + a_j + b_{P+1-i}) - 2.nom$$

$$\beta_1 + \delta_1 = \beta_2 + \delta_2$$

Therefore, $F'_1 = F'_2$

By contradiction, F'_1 can be said to be optimal for the problem.

Second: Proof for the first term; *F*''.

Define: $mean = \frac{a_1 + a_2 + ... + a_P + b_1 + b_2 + ... + b_P}{P}$

The objective function for solution 1 can be expressed as follows:

$$F''_1 = \mathrm{A} + \mathrm{B}_1 + \Gamma + \Delta_1 + \Lambda \quad \text{(A13)}$$

$$F''_2 = \mathrm{A} + \mathrm{B}_2 + \Gamma + \Delta_2 + \Lambda \quad \text{(A14)}$$

Where:

$$\mathrm{A} = \left|(a_1 + b_P) - mean\right| + \left|(a_2 + b_{P-1}) - mean\right| + \left|(a_{i-1} + b_{P-i}) - mean\right| \quad \text{(A15)}$$

$$\mathrm{B}_1 = \left|(a_i + b_{P+1-i}) - mean\right| \quad \text{(A16)}$$

$$\mathrm{B}_2 = \left|(a_i + b_{P+1-j}) - mean\right| \quad \text{(A17)}$$

$$\Gamma = \left|(a_{i+1} + b_{P+2-i}) - mean\right| + \left|(a_{i+2} + b_{P+3-i}) - mean\right| + ... + \left|(a_{j-1} + b_{P+1-j}) - mean\right| \quad \text{(A18)}$$

$$\Delta_1 = \left|(a_j + b_{P+1-j}) - mean\right| \quad \text{(A19)}$$

$$\Delta_2 = \left|(a_j + b_{P+1-i}) - mean\right| \quad \text{(A20)}$$

$$\Lambda = \left|(a_{j+1} + b_{P+2-j}) - mean\right| + \left|(a_{j+2} + b_{P+3-j}) - mean\right| + ... + \left|(a_P + b_1) - mean\right| \quad \text{(A21)}$$

Suppose that the solution 2 improves the solution provided in solution 1, i.e. $F''_2 \le F''_1$. Notice that we only swap the sequence of two items (b_{P+1-j}, b_{P+1-i}) in group *B*.

There are six cases to compare F''_1 with F''_2:

1. $a_j + b_{P+1-i} \le mean$

$$B_1 + \Delta_1 = (a_i + a_j + b_{P+1-i} + b_{P+1-j}) - 2.mean$$

$$B_2 + \Delta_2 = (a_i + a_j + b_{P+1-j} + b_{P+1-i}) - 2.mean$$

$$B_1 + \Delta_1 = B_2 + \Delta_2$$

Therefore, $F''_1 = F''_2$

2. $\max(a_i + b_{P+1-i}, a_j + b_{P+1-j}) \le mean < a_j + b_{P+1-i}$

$$B_1 + \Delta_1 = 2.mean - (a_i + a_j + b_{P+1-i} + b_{P+1-j})$$

$$B_2 + \Delta_2 = a_j + b_{P+1-i} - mean + mean - a_i - b_{P+1-j} = (a_j - a_i) + (b_{P+1-i} - b_{P+1-j})$$

$$(B_1 + \Delta_1) - (B_2 + \Delta_2) = 2(mean - (a_j + b_{P+1-i})) \le 0$$

Therefore, $F''_1 \le F''_2$

3. $a_i + b_{P+1-i} \le mean < a_j + b_{P+1-j}$

$$B_1 + \Delta_1 = mean - a_i - b_{P+1-i} + a_j + b_{P+1-j} - mean = (a_j - a_i) + (b_{P+1-j} - b_{P+1-i})$$

$$B_2 + \Delta_2 = mean - a_i - b_{P+1-j} + a_j + b_{P+1-i} - mean = (a_j - a_i) + (b_{P+1-i} - b_{P+1-j})$$

$$(B_1 + \Delta_1) - (B_2 + \Delta_2) = 2(b_{P+1-j} - b_{P+1-i}) \le 0$$

Therefore, $F'_1 \le F'_2$

4. $a_j + b_{P+1-j} \le mean < a_i + b_{P+1-i}$

$$B_1 + \Delta_1 = a_i + b_{P+1-i} - mean + mean - a_j - b_{P+1-j} = (a_i - a_j) + (b_{P+1-i} - b_{P+1-j})$$

$$B_2 + \Delta_2 = mean - a_i - b_{P+1-j} + a_j + b_{P+1-i} - mean = (a_j - a_i) + (b_{P+1-i} - b_{P+1-j})$$

$$(B_1 + \Delta_1) - (B_2 + \Delta_2) = 2(a_i - a_j) \leq 0$$

Therefore, $F'_1 \leq F'_2$

5. $a_i + b_{P+1-j} \leq mean < \min(a_i + b_{P+1-i}, a_j + b_{P+1-j})$

$$B_1 + \Delta_1 = a_i + b_{P+1-i} + a_j + b_{P+1-j} - 2.mean$$

$$\beta_2 + \delta_2 = mean - a_i - b_{P+1-j} + a_j + b_{P+1-i} - mean = (a_j - a_i) + (b_{P+1-i} - b_{P+1-j})$$

$$(\beta_1 + \delta_1) - (\beta_2 + \delta_2) = 2((a_i + b_{P+1-j}) - mean) \leq 0$$

Therefore, $F'_1 \leq F'_2$

6. $mean < a_i + b_{P+1-j}$

$$B_1 + \Delta_1 = 2.mean - (a_i + a_j + b_{P+1-i} + b_{P+1-j})$$

$$B_2 + \Delta_2 = 2.mean - (a_i + a_j + b_{P+1-j} + b_{P+1-i})$$

$$B_1 + \Delta_1 = B_2 + \Delta_2$$

Therefore, $F''_1 = F''_2$

By contradiction, F''_1 can be said to be optimal for the problem.

Therefore, $F = F'_1 + F''_1$ is the optimal solution for the IP presented in Equations 5.5-5.9.

VITA

Rami Musa was born in Kuwait in December 28th, 1975. He graduated with a Bachelor of Science degree in Mechanical Engineering in 1999 from Jordan University of Science and Technology. After his graduation, he worked as a design engineer in Petra Engineering Inc. and Omrania Associates in Amman, Jordan. In 2001, Rami moved to the United States to pursue his Master's in Industrial Engineering at the University of Cincinnati. He graduated in 2003 with a Master's degree and several published papers in journals and conference proceedings. In the same year, he started his Ph.D. studies at Virginia Tech in the Grado Department of Industrial and Systems Engineering. He worked in the department as instructor, teaching assistant and research assistant. He is a recipient of Pratt and Dover fellowships in 2006 and 2007; respectively. He published several papers in leading journals and conference proceedings. In March 2007, he successfully defended his Ph.D. dissertation. He spent his summers during graduate school working as a graduate intern in major companies such as Trim Systems (Volvo supplier) in Dublin, VA and General Electric (Global Research Laboratory) in Niskayuna, NY.