AN ABSTRACT OF THE THESIS OF

Carlos H. Capps for the degree of Master of Science in Industrial Engineering presented on December 3, 1997. Title: Setup Reduction in PCB Assembly: A Group Technology Application Using Genetic Algorithms.

Redacted for Privacy

Abstract approved: _____
Terrence Beaumariage

Redacted for Privacy

Abstract approved: _____
Brian K. Paul

For some decades, the assembly of printed circuit boards (PCB), had been thought to be an ordinary example of mass production systems. However, technological factors and competitive pressures have currently forced PCB manufacturers to deal with a very high mix, low volume production environment. In such an environment, setup changes happen very often, accounting for a large part of the production time.

PCB assembly machines have a fixed number of component feeders which supply the components to be mounted. They can usually hold all the components for a specific board type in their feeder carrier but not for all board types in the production sequence. Therefore, the differences between boards in the sequence determines the number of component feeders which have to be replaced when changing board types. Consequently, for each PCB assembly line, production control of this process deals with two dominant problems: the determination for each manufacturing line of a mix resulting in larger similarity of boards and of a board sequence resulting in setup reduction. This has long been a difficult problem since as the number of boards and lines increase, the number of potential solutions increases exponentially.

This research develops an approach for applying Genetic Algorithms (GA) to this problem. A mathematical model and a solution algorithm were developed for effectively determining the near-best set of printed circuit boards to be assigned to surface mount lines. The problem was formulated as a Linear Integer Programming model attempting to setup reduction and increase of machine utilization while considering manufacturing constraints. Three GA based heuristics were developed in order to search for a near optimal solution for the model. The effects of several crucial factors of GA on the performance of each heuristic for the problem were explored. The algorithm was then tested on two different problem structures, one with a known optimal solution and one with a real problem encountered in the industry. The results obtained show that the algorithm could be used by the industry to reduce setups and increase machine utilization in PCB assembly lines.

Setup Reduction in PCB Assembly:
A Group Technology Application Using Genetic Algorithms

by

Carlos H. Capps

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed December 3, 1997
Commencement June 1998

Master of Science thesis of Carlos H. Capps presented on December 3, 1997.

APPROVED:

Redacted for Privacy

Co-Major Professor representing Industrial Engineering.

Redacted for Privacy

Co-Major Professor representing Industrial Engineering.

Redacted for Privacy

Head of Department of Industrial and Manufacturing Engineering

Redacted for Privacy

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for Privacy

Carlos H. Capps, Author

ACKNOWLEDGMENT

The work presented here represents the culmination of two years of academic research conducted in the Industrial and Manufacturing Engineering Department at Oregon State University. Many institutions and individuals have played important roles in this research effort.

I would like to take this opportunity to thank the Fulbright Commission and the American Chamber of Commerce and its International Fellowship Program for their generous financial assistance which allowed me to pursue my graduate studies. I would also like to thank the Institute for International Education, in particular Ms. Laurie Stevens who has helped me on many occasions with her knowledge and kindness.

I would also like to express my gratitude to Tektronix Inc., in particular to Mr. Ken Carlson and Mr. Richard Kurschner, for sponsoring this research and for allowing the implementation of this work in the real world.

I have had the good fortune to have interacted with a large number of people who have offered friendship and assistance throughout my study here at Oregon State University. I must, in particular, thank Dr. Terry Beaumariage, who helped me not only with my thesis but also with some nonacademic matters, always demonstrating knowledge, professionalism and friendship. I would also thank Dr. Brian K. Paul who was instrumental in obtaining a "real world" research project. Very special thanks to Dr. Ed Mcdowell, whose knowledge of statistics was extremely helpful. His interest and humor shown throughout the study actually made working with numbers a challenge and a pleasure. Finally, I would like to thank Dr. Logen Logendran who sparked my initial interest in Genetic Algorithms and whose class provided helpful ideas in group scheduling and cellular manufacturing.

I am also grateful for having Thomas Kretz as a friend and partner during this time in Graduate School. He has stood as a model of diligence and kindness. His suggestions and comments on Genetic Algorithms were valuable and constructive contributing considerably to the success of this work. It is a great honor to have had the opportunity to work with someone like him and to have a friend like him.

I would like to extend a heartfelt thank you to each member of my family, who have all believed in me and encouraged me throughout my every endeavor.

It suffices to say that nothing would be possible without the sacrifices and support of my parents, Isabel and Alberto Capps (in memoriam), who taught me the value of education. In particular to my mother, whose continual support and encouragement was a constant strength throughout the entire research.

Most of all I would like to express my deepest gratitude to my wonderful wife and best friend, Luciene, for her devotion and unwavering support through all these years. Her encouragement, sensitivity, knowledge and especially her deep kindness have made it possible to complete this thesis. This thesis is lovingly dedicated to her and to my parents.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDIX FIGURES

LIST OF APPENDIX FIGURES (Continued)

LIST OF APPENDIX FIGURES (Continued)

LIST OF APPENDIX TABLES

LIST OF APPENDIX TABLES (Continued)

Setup Reduction in PCB Assembly: A Group Technology Application using Genetic Algorithms

INTRODUCTION

Fierce competition in the electronic industry is causing profound changes. An example is the shortening of the life-cycle of products which forces customization or the introduction of multiple options for the same basic product. The overall effect of these trends is requirements for denser geometries, faster design cycles, stricter process control, greater reliability, alternative materials and components, and lower total cost. In the production environment, the result is the reduction of lot sizes aiming at reducing lead times and inventories, and at increasing flexibility. Thus, it is apparent that these trends force changes in all elements of the total electronic products design, fabrication, and assembly process.

Printed circuit boards (PCB) form the essential part of the electronics industry. The printed circuit continues to be the basic interconnection technique for electronic devices. Virtually every packaging system is based upon this process and undoubtedly will continue to be in the near future. Its technology continues to grow and change. One of the most important changes in the past few years was the development of Surface Mount Technology (SMT). It has emerged as a major interconnection and packaging technology, introducing revolutionary changes in the electronic industry.

For some decades, production in the electronics industry, especially the assembly of PCBs, had been thought to be an ordinary example of mass production systems. However, with the shortening of the products' life cycle, the market demands manufacturers to produce a large variety of PCBs with small lot size. Currently PCB manufacturers deal with a very high mix, low volume production environment where a lot size of one is not abnormal.

In such an environment with very small batch production and wide product diversity, setup changes happen very often. These setups account for a large part of the production time causing an immense idle time of the machines. Therefore, setup reduction is critical to shorten the production lead time and increase the machine utilization, and it is the objective of this work.

PCB assembly problems

Modern PCB production typically uses computerized machines to automatically place electronic components onto boards. A variety of component types, including chips, resistors, capacitors, ball grid arrays (BGA), etc. and many other components which differ by package type, are typically required by each board.

Due to the continuous development in the electronics industry, the number of components per board has been greatly increased in the last decade. It is not unusual that a single circuit board would require between 50 and 200 component types.

PCB assembly machines have a fixed number of component feeders which supply the components to be mounted. Depending on component size a single feeder can supply one or two components (single or double feeder). Usually high speed placement machines (HSPM) can hold all the components for a specific board type in its feeder carrier. If all the required components are not available on the feeder carrier for assembling a PCB, a new setup of components will be required.

For HSPM most of the CNC programming software available attempts to optimize the travel distance of the head when placing components as well as optimizing the travel distance of the carrier when loading a component. As this is done individually for each board type, the software determines different feeder positions in the carrier for each board. Therefore, since each component type on a PCB must occupy one feeder on the carrier, when the PCB batch is changed, the component feeders must be re-setup.

This setup time is proportional to the number of component feeders which have to be replaced or moved when changing board types. Therefore, the sequencing of boards is a critical factor for the setup reduction.

Although HSPM have rapid assembly rates, their setup times are rather long. A typical machine is capable of assembling thousands of components per hour while its setup time (loading and unloading components from the machine due to changeover of PCB types) may take about an hour. When used in a high volume environment this assembly rate leads to high efficiency. However when in a low volume, high mix environment, the assembly of small batches suffers from inefficiency caused by long setup times. Therefore, it is clear in the latter case that any reduction in setup times allows increases in machine productivity and reduction in manufacturing lead time.

Furthermore, in order to stay competitive, manufacturers have to constantly update their manufacturing systems. Frequently this is not a feasible option and manufacturers have to keep equipment with different technological levels and characteristics as part of the same system. Consequently, the assignment of boards must consider specific elements of each line so as to fulfill throughput and technological requirements of each board.

Therefore, production control of this process deals with several problems, including selection of board types assigned to each line to fulfill requirements and reduce setup in the overall mix, allocation of component types to each machine so as to maximize utilization and reduce make span, and determination of board sequencing to reduce setup times.

These are some of the main problems that the electronic assembly industry faces at this time. For continued success these problems have to be studied and improved solutions developed. Any improvement can increase competitiveness.

Significance of the Research

The high cost of PCB assembly machines justifies careful planning and control of their operations. High assembly rates and the slow setup times call for particular attention to the setup issue. Traditionally, two main approaches are employed for reducing the overall setup time needed for production:

- Minimizing the number of times each PCB type is loaded onto the machine;
- Minimizing the number of times each component type is loaded on the machine;

The first approach simply tries to increase the lot sizes, consequently reducing the setup frequency. However, in many cases due to economic reasons, the lot size is limited and cannot be increased. Increasing the lot size also means enlarging the work in process (WIP) inventory associated costs.

The second approach is essentially based on Group Technology (GT) concepts. Products are classified into groups using similar components, for which production sequences can be developed. Thus, PCBs are sequenced such that a job subsequence requires the same component types, eliminating much of the setup between them.

The basic problem in GT is the determination of groups. Massive research has been devoted to cluster analysis and similarity measures. These methods are used in the process of partitioning a set of items and the corresponding set of machines (used for the processing of these components) into subgroups. Each subgroup of items is called a 'family' and the set of machines used to process these items is called a subsystem or 'cell'.

For the PCB assembly process the cell formation problem requires clustering PCBs into groups that require a similar set of components and assigning each group to an automated assembly line. The objective is to minimize the total setup time. Hence, two boards are similar if the mix of their components is similar.

These techniques usually represent the PCB-component relationship with a PCB-component incidence matrix and then apply algorithms or heuristics to perform the grouping. They generally allow for a more global setup minimization, but usually do not ensure a sufficient load balance, since the workloads of groups are not considered by commonly used grouping procedures. Also, they do not consider the possibility of trading boards from different lines in the grouping procedure which would allow better results. They focus on the minimization of changeovers within the groups which does not guarantee a global minimization since the setup between groups can be increased depending on the results. Finally, the application of these techniques in large and realistic problems has proven problematic in terms of computational time.

Several studies focus on the optimal utilization of HSPM by emphasizing the reduction of placement cycle-time. They tend to neglect the fact that the setup requirement for a job is dependent upon the sequence of jobs. Although the machine state is an important decision that affects the system performance, it should be determined simultaneously with the product sequence since there is a tradeoff between setup time and processing time. Optimal utilization of resources can be achieved only by simultaneously reducing the processing time required for a PCB and the setup time encountered between PCB changeovers.

Other studies try to achieve setup reduction through balancing of workloads and control of assembly sequences. They are applied to short term scheduling of production systems aiming at minimizing setup times and maximizing workload balance. However, they are often designed to perform their optimization on a mix which is limited both in space and time. This may assure local improvements but it does not add much to the overall problem since only a small piece of the problem is being investigated.

Therefore, although much research work has been devoted to process optimization on PCB assembly systems, it is felt that more research is still needed in the area of board sequencing and setup reduction.

It is believed that the PCB assembly problem has not yet been realistically approached since one of its critical factors has not been considered: "The assignment of boards to lines". The PCB assembly problem can be divided in the following areas:

- Assignment of boards to lines;
- Assignment of component to machines;
- Production planning (sequencing);
- Subcontracting of bottleneck boards;

The first area is concerned with improvement of the overall system's performance. The goal here is to find the best allocation for boards according to the technological, process, and demand constraints so as to improve some performance measure. The second is concerned with improving machine utilization by determining the best distribution of components to machines in the manufacturing line. The third is concerned with sequencing boards to reduce setups and improve line performance. Finally, the last one is concerned with choosing the boards that have to be subcontracted in order to improve the system's performance. Some of these areas can be approached simultaneously, greatly increasing the problem's complexity.

As discussed before, many studies have been accomplished in the past to investigate the part family-machine cell formation problem. Most of them focus on the improvement of productivity through reducing intercell moves. Kusiak and Chow (1988) present a complete list of references on this subject. These works, although important, can not be applied to the PCB assembly problem since production is arranged in a group layout (flow line cell) and there are no moves between cells. Therefore, a specific approach has to be developed for the part family-machine cell formation problem in PCB assembly.

It is proven that setup is the most important factor in reducing manufacturing productivity on PCB assembly. It is also known that setup time is proportional to the number of component feeders which have to be replaced when changing board types. Hence, in order to reduce manufacturing costs, an approach must consider grouping

boards attempting to reduce the number of feeder changes. Since this is a sequence dependent problem and therefore very dynamic, the groups have to assure that when sequenced they yield setup reduction.

Consequently, a real improvement is not guaranteed if boards are grouped only regarding similarity by components. Demand is a variable that determines the need for a certain board and does not assure that boards belonging to the same group will be scheduled at the same time. Therefore, boards have to be grouped according to their similarity and assessment of possible sequences which might occur in a real environment.

With that purpose the approach proposed here focuses on assigning boards to lines attempting to reduce manufacturing costs through reducing setup. Sequencing is considered and groups are formed so as to achieve minimum setups within groups, minimum setups between groups, and an overall setup reduction even when boards are scheduled differently within a group.

The production environment that can benefit most from the implementation of this method is a high-mix, low-volume production environment.

Overview of Approach

First, the part assignment problem will be formulated as a Linear Integer Programming attempting to setup reduction and considering sequence dependence. Then, the complexity of the problem will be established and a heuristic based approach using Genetic Algorithms will be developed in order to search for near optimal solutions. This heuristic will be tested experimentally and the result will be applied in a realistic problem.

LITERATURE SURVEY

Printed Circuit Boards (PCBs)

Printed circuit boards are dielectric substrates with metallic circuitry photochemically formed upon that substrate. There are three major classifications:

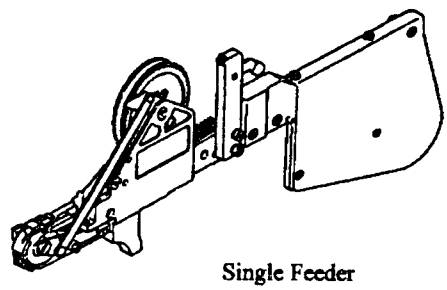
- *Single sided boards*: Dielectric substrate with circuitry on one side only. There may or may not be holes for components and tooling drilled into them.
- *Double sided boards*: Dielectric substrate with circuitry on both sides. An electrical connection is established by drilling holes through the dielectric and plating copper through the holes.
- *Multilayer boards*: Two or more pieces of dielectric material with circuitry formed upon them are stacked up and bonded together. Electrical connections are established from one side to the other, and to the inner layer circuitry by drilled holes which are subsequently plated through with copper.

PCB Technologies

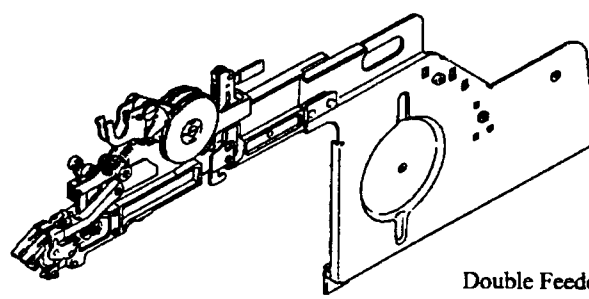
Originally, electronics assembly used point-to-point wiring and no dielectric substrate. The first improvement came with the semiconductor packages using radial leads. They were inserted in through holes of the single-sides circuit boards already used for resistors and capacitors. Through-hole printed wiring technology is no longer adequate to meet the needs of high-performance electronic assemblies.

Surface mount technology (SMT) is used to mount electronic components on the surface of printed circuit boards. SMT allows production of more reliable assemblies at reduced weight, volume, and cost. The size reduction on surface mount components, a direct function of lead pitch (distance between the centers of adjacent leads), allows the

assembly of components on either side of the boards. Routings of traces is also improved because there are fewer drilled holes, and these holes are smaller. The size of a board is reduced and so is the number of drilled holes. This fact, associated with the savings in weight and real-estate, and with electrical noise reduction resulted in widespread usage.



Single Feeder



Double Feeder

Figure 1 - Typical Feeders used on HSPM

PCB Assembly

Automatic assembly of printed circuit boards is a process in which components are mounted on (or inserted into) the PCB according to its electrical design. The production process is composed of a placement/insertion mode and a soldering mode, followed by quality assurance processes. The equipment is composed of an insertion

head and a feeder mechanism upon which reels are mounted, each containing components of the same type. The placement/insertion has to be arranged in a certain sequence so as to reduce the overall placement time. Figures 1 and 2 (Panasert MVII Technical Guide, 1995) show the typical feeders used on High Speed Placement Machines (HSPM) and a typical High Speed Placement Machine, respectively.

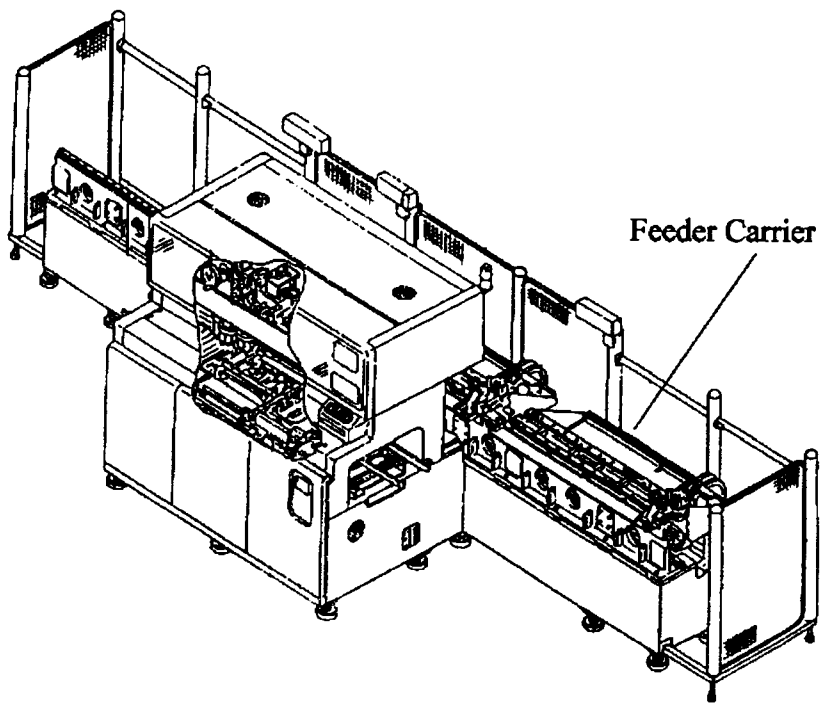


Figure 2 - Typical High Speed Placement Machine

The typical process sequence for the total SMT is shown in figure 3. First solder paste is screened onto the bareboard, components are placed, and the assembly is baked in a convection or infrared oven to heat the components to a certain temperature. Finally, the assembly is reflow soldered in the same oven and then solvent cleaned. For double-sided boards, the board is turned over and the process sequence is repeated. Although the solder joints on the top side of the board are reflowed again, the underside components are held in place by the surface tension of previously reflowed solder.

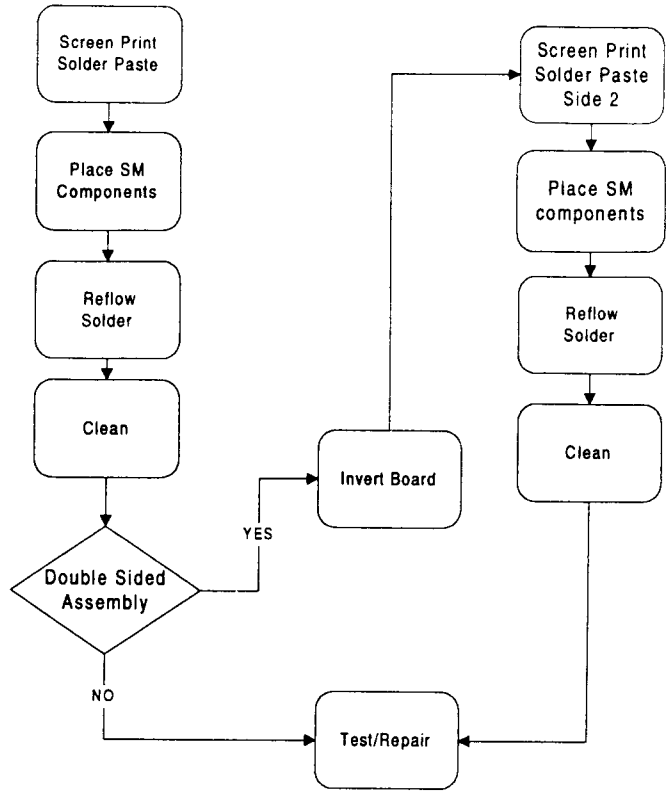


Figure 3 - Typical process flow for total surface mount

The typical process sequence for mixed technology is shown in figure 4. First the adhesive is dispensed and the Surface Mount (SM) components are placed and reflow soldered as it is for the total SMT process. The assembly is then turned over and the through hole components are manually inserted. Next the assembly is wave soldered, cleaned, and tested.

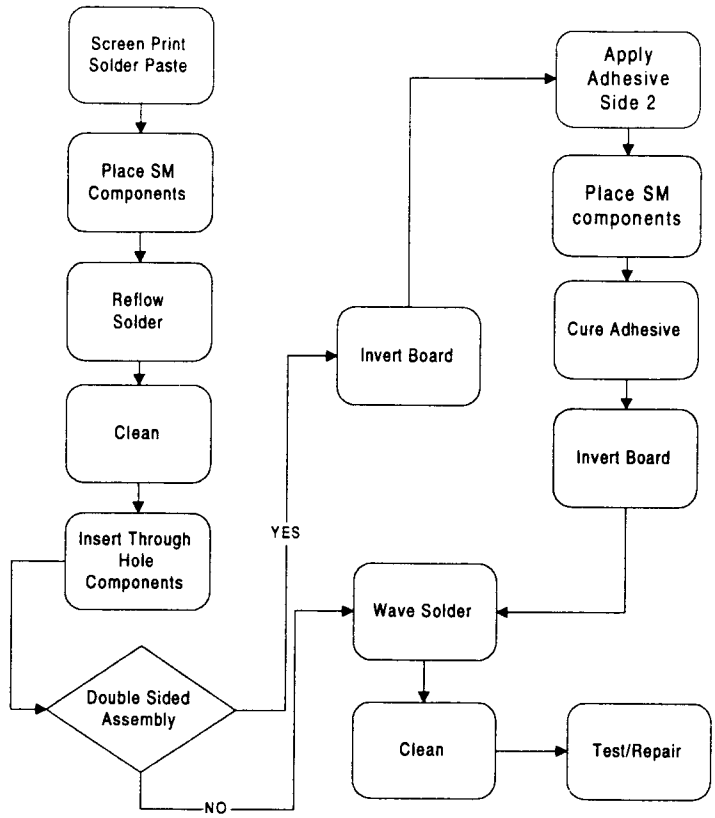


Figure 4 - Typical process flow for mixed technology

Group Technology

Group Technology (GT) has been recognized as the key to improve productivity, material-handling, management and control of a typical batch-manufacturing system (Harhalakis, Nagi and Proth 1990). This has profound implications on the profitability and overall operational efficiency of a manufacturing organization. GT can also simplify the design and process planning of new products by taking advantage of similarities in part design and manufacturing characteristics.

The basic idea of GT viewed solely from a manufacturing viewpoint is the decomposition of the manufacturing system into subsystems, by classifying parts into families, and machines into machining cells, based on the similarity of part

manufacturing characteristics. Parts that have to undergo similar operations and require a given set of machines for these operations are grouped together. These machines are in turn grouped into a machine cell, thus forming a subsystem or cell.

The basic problem in group technology is the establishment of item groups that could utilize the priorities of similarity between them during the manufacturing process (Shiko 1992). The term 'item' can represent almost all the constituent elements in a manufacturing system, such as designed parts, process plans, machine tools, jigs, fixtures, tools, manufacturing processes and the like.

Traditional clustering algorithms have been applied to group technology for some time and are usually based on serial processing techniques. On these algorithms, the machine-component group analysis problem, may, in the simplest form, be expressed as that of determining, by a process of row and column exchanges of the matrix, a conversion from a haphazard pattern of entries into an arrangement whereby the 'one' entries are contained in mutually exclusive groups arranged along the diagonal of the matrix (King 1980).

The machine-component matrix is the main input to most machine component grouping models (Seifoddini and Djassemi 1996). It is an M x N matrix with zero/one entries. The presence or absence of 'one' entry in row i and column j of the matrix indicates the operation, or lack of operation, of part j on machine i, respectively. When natural machine-component groups exist in a production system, the rearrangement of parts and machines in the corresponding machine-component matrix generates a block diagonal form in which 'one' entries are concentrated in blocks along the diagonal of the matrix (Burbidge 1975). These blocks correspond to machine-component groups which are used to form a cellular manufacturing system.

Genetic Algorithms

Genetic algorithms (GA) are practical, robust optimization and search methods rooted in the mechanisms of evolution and natural genetics. They were first proposed by Holland (1975) who demonstrated how the evolutionary process can be used to solve problems by means of a highly parallel technique.

Genetic algorithms transform a population of individual objects, each with an associated value of fitness, into a new generation of the population, using the Darwinian principle of survival and reproduction of the fittest as the selection mechanism, and crossover and mutation as search mechanisms (Davis, 1991). Only the fittest individuals survive and reproduce. The genes of the fittest survive, while the genes of weaker individuals die out.

Genetic algorithms operate on encoded representations of the solutions, equivalent to the genetic material of individuals in nature, and not directly on the solutions themselves. Usually the solutions are encoded as strings of bits from a binary alphabet. Recombination of genetic material in GAs is simulated through a crossover mechanism that exchanges portions between strings. Another operation, called mutation, causes sporadic and random alteration of the bits of strings. Following the four major steps required to use GAs on fixed-length character strings (Koza, 1994):

1. *The representation scheme.*

   Mapping each possible point in the search space of the problem as a particular fixed-length character string or chromosome (determining the string length and the alphabet size, usually the alphabet is binary) and each such chromosome as a point in the search space of the problem.

2. *The fitness measure.*

   It has to be capable of evaluating any string that it encounters in any generation of the population.

3. *The parameters and variables for controlling the algorithm.*

   The population size, the maximum number of generations to be run, and quantitative and qualitative control variables that must be specified in order to run the GA.

4. *A way of designating the result and a criterion for terminating a run.*

   It consists of either satisfying a problem-specific success predicate or completing a specified maximum number of generations to be run.

Following the steps in executing the GA operating on character strings:

1. Randomly create an initial population of individual fixed-length character strings.
2. Iteratively perform the following substeps on the population of strings until the termination criterion has been satisfied:
   (a) Assign a fitness value to each individual in the population using the fitness measure.
   (b) Create a new population of strings by applying the following three genetic operations. The genetic operations are applied to individual string(s) in the population selected with a probability based on fitness (with reselection allowed).
      (i) Reproduce an existing individual string by copying it into the new population.
      (ii) Create two new strings from two existing strings by genetically recombining substrings using the crossover operation at randomly chosen crossover point.
      (iii) Create a new string from an existing string by randomly mutating the character at one randomly chosen position in the string.
3. Designate the string that is identified by the method of result designation (the best-so-far) as the result of the GA for the run. This result may represent a solution (or an approximate solution) to the problem.

Genetic Algorithms are probabilistic algorithms. Probabilistic steps are involved for creating the initial population, selecting individuals from the population on which to perform each genetic operation (e.g., reproduction, crossover), and choosing a point (crossover point or a mutation point) within the selected individual at which to perform the selected genetic operation. Additional probabilistic steps are often involved in measuring fitness.

Previous Work on Group Technology

Chandrasekharan and Rajagopalan (1987) developed an algorithm for concurrent formation of part-families and machine-cells in group technology. The formation of part-families and machine-cells was treated as a problem of block diagonalization of the zero-one matrix. A grouping efficiency (GE) measure was developed to evaluate the efficiency of block diagonal matrices. It is defined as:

$$GE = qE_1 + (1-q)E_2$$

where:

$$E_1 = \frac{\text{Number of ones in the diagonal blocks}}{\text{Tot. number of elements in the diagonal blocks}}$$

$$E_2 = \frac{\text{Number of zeros in the off-diagonal blocks}}{\text{Total number of elements in the diagonal blocks}}$$

The selection of q (weighting factor) is arbitrary and according to the designer of the measure (Kumar and Chandrasekharan, 1990) the range of values for this measure was limited to 75-100%. That means that even when there is a large number of exceptional parts, the grouping efficiency of the machine-component matrix is at least 0.75.

Later on, the authors proposed another grouping measure to overcome the problems of the selection of q (Kumar and Chandrasekharan, 1990), the grouping efficacy measure (GC). It is defined as:

$$GC = qE_1 + (1-q)E_2$$

in which:

$$q = \sum_{r=1}^{K} \frac{M_r . N_r}{m.n}$$

$$E_1 = \frac{e_o}{\sum_{r=1}^{K} M_r . N_r}$$

$$E_2 = 1 - \frac{e_o}{m.n - \sum_{r=1}^{K} M_r . N_r}$$

where:

K = number of blocks

M_r = number of rows in the rth block

N_r = number of columns in the rth block

e_o = number of ones in the diagonal blocks

m = number of rows in the binary matrix

n = number of columns in the binary matrix

Group efficacy overcomes the problem of grouping efficiency by incorporating the matrix size into the calculation of the measure. It also provides a quantitative basis for calculating the weighting factor q. Hsu (1990) has shown that neither efficiency nor

group efficacy is consistent in predicting the performance of a cellular manufacturing system based on the structure of a corresponding machine-component matrix.

The approach above can be used in determining groups of components and boards. A matrix indicating which components are used in which boards can be done and the algorithm can be applied. Besides the problem that it does not thoroughly represent the performance of the system, the approach does not incorporate production volume and processing times in the evaluation of the goodness of solutions. Therefore, boards that have a very low demand (few batches per year) can be grouped with boards with high demand (several batches per year) and the outcome on the overall set-up times will be ineffective. Furthermore, in realistic problems with a large number of components and boards this approach is computationally inefficient.

Maimon and Shtub (1991) developed a mathematical programming formulation for grouping a set of Printed Circuit Boards (PCB). The assembly machines are configured for each group, thus saving set-up time. The formulation is defined as:

$$Min\,\tau$$

Subject to:

$$\tau \geq \sum_{g \in G} \left(\sum_{i \in I} S_i y_{ig} + \sum_{j \in J} S_j x_{jg} \right)$$

$$Fij \leq M \sum_{g \in G} x_{jg} y_{ig} \quad \text{for all } i \text{ and } j$$

$$\sum_{i \in I} y_{ig} \leq L$$

where:

τ the time spent on set-up of PCBs and components during the planning period.

$i \in I$ index set of component types, i=1...M.

S_i set-up time for a single component type i.

$j \in J$ index set of PCB types, j=1...N.

S_j set-up time for PCBs type j (i.e., the time to load on the machine and unload the PCB itself.

F_{ij} number of components of type i required for PCB j.

L capacity (number of component types that can be loaded simultaneously) of the assembly machine.

$g \in G$ index set of groups of PCBs and components, g=1...K.

x_{jg} = 1 if PCB j is loaded with group g

0 otherwise

y_{ig} = 1 if component type i is loaded (assembled) with group g

0 otherwise

The approach above assumes that boards were already assigned to machines. A heuristic is proposed using matrix diagonalization (King and Nakornchai, 1984). The columns of the matrix represent component contents of PCBs, where adjacent PCBs are similar to each other. The major concern is to reduce set-up through the ordering of boards in the matrix. The drawback (like in Chandrasekharan and Rajagopalan, 1987) is that it does not incorporate process and production information on the evaluation of the goodness of the solutions. Therefore, in order to produce the best performance, the groups formed have to be sequenced as in the solution achieved with the heuristic. Thus, when in a realistic situation where boards have to be scheduled according to demand and

lot sizes, the outcome may not have good results in the overall setup times. Furthermore, the heuristic can only be used in small problems due to its computational time.

Shtub and Maimon (1992) suggested a general approach based on cluster analysis and measure of similarity between PCBs. The PCB members of each subgroup share common components in such a way that component set-up is saved and the utilization of the assembly machine is improved. The group formation is based on the Jaccard similarity index (Sokal and Sneath - 1963) defined for each pair of PCB types j, l as follows (where η_j is the set of components required for PCB type j):

$$SI_{j,l} = \frac{|\eta_j \cap \eta_l|}{|\eta_j \cup \eta_l|}$$

The numerator is the number of components required for both PCBs j and l, the denominator is the number of components that are required for at least one of the PCBs j and l.

PCBs are added to a group in order of the similarity between the PCB considered and that group, subject to capacity constraint (the number of component types that can be loaded on the machine) and any performance measure defined by the user. Termination is achieved when all PCBs are assigned to groups in such a way that the set of components required by each PCB is covered by the sets of components associated with the groups where the PCB is a member. Although using a different approach than the block diagonalization, the value of the work is approximately the same. Here, the authors use a similarity coefficient to measure the 'bond energy' between boards. As the previous ones discussed where production volumes and processing times are not taken into account, the disadvantage of this approach is that sequencing is not considered and in order to achieve the minimum setup, the boards have to be scheduled as they are grouped. This is a very fixed solution which does not help much in a realistic and

dynamic environment. Moreover, the heuristic proposed is not feasible for realistic problems due to computational time.

Seifoddini and Djassemi (1996) developed a new grouping measure for evaluation of machine-component matrices, the 'quality index' (QI), which incorporates production volume and processing times. It is calculated as the ratio of the intercellular workload to the total plant's workload. Simulation modeling estimating average flow time and average in-process inventories was used to determine the relationship between values of grouping measures and the performance of the corresponding cellular manufacturing system. The intercellular workload (ICW) is defined as:

$$ICW = \sum_{l=1}^{K}\left[\sum_{i=1}^{M} X_{il}\left(\sum_{j=1}^{N}(1-y_{jl})Z_{ij}.V_j.T_{ij}\right)\right]$$

where:

X_{il} = 1 if machine i is assigned to machine cell l

0 otherwise

Y_{jl} = 1 if part j is assigned to machine cell l

0 otherwise

Z_{ij} = 1 if part j has operations on machine i

0 otherwise

V_j = production volume for part j

T_{ij} = processing time of part j on machine i

K, M, and N = number of machine cells, machines, and parts, respectively

The total plant workload (PW) can be calculated as:

$$PW = \sum_{i=1}^{M} \sum_{j=1}^{N} Z_{ij} . V_j . T_{ij}$$

The quality index (QI) for a block diagonal machine-component matrix calculated is:

$$QI = 1 - \frac{ICW}{PW}$$

Although the study shows that grouping measures when properly defined, can predict the performance of a cellular manufacturing system, it does not incorporate setup times between parts within the groups and between groups in the measure proposed, therefore weakening the evaluation of block diagonal forms. Furthermore, this approach, as the ones discussed before, uses block diagonalization which is not feasible for realistic problems due to the computational time.

Bhaskar and Narendran (1996) address the problem of grouping PCBs on a single machine, with the objective of minimizing the total set-up time subject to the capacity constraint. The objective function is defined as:

Minimize:

$$\sum_{g=1}^{G} \left(\sum_{i=1}^{N} S . x_{ig} + \sum_{j=1}^{M} s . y_{ig} \right)$$

where:

x_{ig} = 1 if PCB i is assigned to group g

0 otherwise

y_{ig} = 1 if component j is assigned to group g

0 otherwise

S = PCB setup time

s = component setup time

N = number of PCB types

M = number of component types

G = number of PCB and component groups

A heuristic based on a maximum spanning tree approach is developed for grouping the PCBs. A network is constructed with PCBs as nodes and the similarities between them as the weights of the arcs. PCBs are grouped on the basis of the similarity between them. A new measure of similarity, called the cosine similarity coefficient, is introduced. Considering each column as a vector in an M-dimensional Euclidean space, the similarity between two PCBs i and j is the cosine of the angle between the pair of vectors that represent the two PCBs. It is defined as:

$$S_{ij} = \cos(\theta_{ij}) = \frac{(\bar{i}.\bar{j})}{|\bar{i}|.|\bar{j}|}$$

It can be seen that the formulation does not reflect the fact that setups are sequence dependent, assuming that any time a board is assigned to a group it has a fixed setup time (PCB setup and components setup). Also, although the coefficient gives a discriminating power to the heuristic proposed, it does not deal with the production volumes and run times, therefore it is not consistent enough to determine the groups. Finally, the approach proposed can be applied only on small problems due to computational time.

Hashiba and Chang (1991) presented a method to reduce setups for PCB assembly. The problem is first formulated as an integer programming model focusing on

finding the sequence which gives the minimum setup times (minimum changeover of components). Next, a three-step approach is taken due to the enormous computation time required when applying the model.

Grouping PCBs is the first step and it is based on the component commonality among different PCB types. The method centers on minimizing the Hamming distance (i.e. number of different components between the PCB and the group) between a PCB and a group. The goal is to create groups where no setup is necessary when changing from one PCB type to another. Consequently, the effective number of PCB types can be reduced from the number of real PCB types to the number of PCB groups. This is done through an exhaustive non-decreasing heuristic function.

The second step is to determine the assembly order of PCB groups. Since no setup is needed within a group, the assembly order in a group is arbitrary. On the other hand, the assembly order among PCB groups has significant effect on the number of setups. This problem is treated as a Traveling Salesman Problem where the distances are the number of components that have to be changed for moving from one PCB group to another. Finally, the third step deals with the assignment of component types to feeders in order to minimize the total setups. It considers the possibility of a group having less component types than the carrier capacity. Therefore, some component types of the preceding setup could be unchanged.

The disadvantage of this approach is that it does not contemplate the run times. Therefore, depending on the number of components of each type assigned to each machine, the solution can result in an unbalanced sequence creating bottlenecks, increasing work in process, and consequently increasing the makespan. Also, the grouping of boards does not look for carrier constraints (e.g. size of components) when assigning components to the carrier. Thus the solution produced may not be feasible. Furthermore, in a realistic problem, the search for groups with no setup, due to the immense number of component and PCB types, will result in very small groups which will have little effect on the overall problem. This exhaustive search might also be

inappropriate for problems with realistic size. Finally, the TSP approach is just a rough approximation since the number of setup changes depends not only on the setup directly before the present setup, but also on all the preceding setups.

Carmon, Maimon and Dar-El (1989) developed an algorithm for the operation planning of PCB assembly machines. The model proposes a different production method, called the group set-up (GSU) method. The main idea behind GSU is that PCBs are assembled in two stages. In the first stage, the common components (i.e. components that are shared among product types in the group) are set up on the machines, once only for the whole group, and are assembled onto their respective PCBs. The next stage requires the separate set-up and assembly of the remaining components on each product. Theoretically, the GSU method always generates least total setup time production plans. However, in real factories, this method is difficult to implement because it requires complete control on the production schedules and when lot sizes are fairly large, the long production makespan generated may be fatal. Also, with large batches the solder paste time constraint may be exceeded causing quality problems or even the complete rejection of the PCB. This approach will greatly increase the work in process (WIP), therefore complicating WIP control and increasing manufacturing costs. Moreover, this approach results in double-handling of the boards, increasing the potential for quality problems related to handling, such as electrostatic discharge.

The problem of grouping PCBs and components is very similar to the one of grouping parts and tools. The latter was addressed by Ventura, Chen and Wu (1990). The authors developed a mathematical formulation for the Flexible Manufacturing Systems part-tool grouping problem and proposed a solution procedure to determine the optimal solution. Given an m x n non-negative matrix where m is the number of rows representing parts, n is the number of columns representing tools, and an element, a_{ij} , denotes the processing time of part i with tool j. The problem is first formulated as an integer programming model focusing on maximizing the total processing time in k groups (which is equivalent to minimizing the interdependencies among the groups). In other words, the part-tool matrix must be rearranged in a way that the resultant matrix

has a block diagonal form where each block represents a group (batch). Although clustering part types and associated tools into groups for batch set-ups is an essential step of the planning process, the formulation does not consider the tool magazine capacity constraints. This might be a problem when assigning the tools in each family to the machines for system set-up (machine loading problem).

Askin, Dror and Vakharia (1994) addressed the problem of minimizing the makespan for assembling a batch of boards with a secondary objective of reducing the mean flow time. The work focuses on the problem of allocating components to surface-mount placement machines and sequencing boards through a multiproduct, multi-machine cell without precedence constraints. The approach adopted is that of grouping boards into production families, allocating component types to placement machines for each family, dividing families into board groups with similar processing times, and the scheduling of groups. A complete setup is incurred only when changing over between board families.

The authors chose to minimize a combination of maximum processing time imbalance within groups (δ_1) and maximum machine work load (δ_2). The open-shop scheduling algorithm is optimal for a group if, at each machine, processing times for all boards in the group are the same. Furthermore, a lower bound on makespan is given by the maximum work load assigned to a machine. The models is formulated as follows:

$$\text{minimize } Z = \alpha\delta_1 + (1-\alpha)\delta_2 \qquad (1)$$

subject to

$$p_{ij} = \sum_{k=1}^{K} t_{jk} Y_{ik} \quad \text{for all } i \text{ and } j \qquad (2)$$

$$\sum_{j=1}^{J} p_{ji} \le \delta_2 \quad \text{for all } i, \qquad (3)$$

$$\bar{p}_{gi} = \frac{\sum_{j=1}^{J} X_{jg} p_{ji}}{\sum_{j=1}^{J} X_{ig}} \quad \text{for all } g \text{ and } I, \qquad (4)$$

$$p_{ji} - \sum_{g=1}^{G} \bar{p}_{gi} X_{jg} \leq \delta_1 \quad \text{for all } j \text{ and } i, \qquad (5)$$

$$p_{ji} - \sum_{g=1}^{G} \bar{p}_{gi} X_{jg} \leq -\delta_1 \quad \text{for all } j \text{ and } i, \qquad (6)$$

where:

x_{jg} = 1 if PCB j is loaded with group g

0 otherwise

y_{ig} = 1 if component type k is assigned to machine i

0 otherwise

Constraint (2) sets processing time for each board on each machine as the sum of component mounting times for the components used by the board that are placed on that machine. Constraint (3) fixes the maximum machine work-load factor. Constraint (4) determines the average processing time on machine i for group g. This is used as a basis for measuring imbalance. Constraints (5) and (6) force δ_1 to be larger than any difference between a board's processing time on a machine and its group average. In addition, the model also ensures that each component is assigned to a single machine, that the feeder slot capacity is not exceeded and that each board is assigned to exactly one group.

Three heuristic approaches are proposed. The first, Component-Assignment/Work-Load Balancing Algorithm (CAWB), is primarily interested in balancing work load between machines. Once components are assigned, similarity of processing time on each machine is used to join boards into groups. Groups then

constitute the basic entity to be scheduled for the open-shop scheduling problem. The second, Work-Load Balancing Algorithm with Shortest Total Processing Time (WBASPT), here the assignment of components to machines was performed identically to the CAWB. However, boards were not grouped for scheduling. Instead, the entire set of boards was considered as a single queue. The individual boards in queue were ordered in increasing order of total processing time on all machines. Whenever a machine becomes available, the first board in the queue list that required that machine is loaded. The third heuristic, Natural Board Subfamily Algorithm (NBSA), uses the natural component composition of boards as the basis for forming subfamilies. The subfamilies are formed based on total board dissimilarity. Then the components are assigned to machines in order to enhance similarity within subfamilies subject to an overall machine work-load balancing constraint. The disadvantage of this approach is that it does not consider technological precedence constraints on component placements, which makes the approach somewhat unrealistic. Also, the model presented assumes that the placement times do not depend on the specific location of component feeders on the machine. Therefore, the solution obtained may increase the run time in some of the machines resulting in an imbalance of the cell.

Stecke and Kim (1991) developed a mathematical programming procedure that selects part types to be machined together over the upcoming time period. It is a flexible approach to short-term production planning which selects mix ratios independent of requirements to maximize production or utilization. The model assumes the demand for part types (or for a subset) is independent in order to determine the relative mix ratios at which a set of part types could be machined together. A disadvantage of this approach is that it might cause a significant increase in the costs associated with inventory, since the model focuses only on the improvement of machine utilization. Also, the environment chosen here with independent demand makes the model very unrealistic.

Tang and Denardo (1988) developed an heuristic approach for the job scheduling problem for a flexible manufacturing machine in the metal working industry. Assuming that each of N jobs must be processed on a machine that has positions for C tools, but no

job requires more than C tools. The problem consists of finding the sequence in which to process the jobs and the tools to place on the machine before each job is processed. First the KTNS (keep Tool Needed Soonest) policy is constructed in order to determine the set of tools to be placed on the machine at each instant so that the total number of tool switches is minimized. The KTNS has the following properties:

1. At any instant, no tool is inserted unless it is required by the next job.
2. If a tool must be inserted, the tools that are kept (not removed) are those needed the soonest.

According to the authors it is possible to get an optimal mix for a given job sequence if the KTNS policy is used. Second the optimal job schedule can be determined by solving the tool replacement problem for each job schedule based on the assumption that the tool replacement problem is solved optimally for every job scheduled. As the problem is proven to be NP-complete a three-step heuristic was develop for finding local optimums for the job scheduling problem. The first step considers a complete graph with N nodes, each of which representing a job. The length of each arc between any two nodes represents the number of tool switches incurred when this pair of jobs is processed consecutively. The length of the shortest Hamiltonian path is a lower bound on the number of tool switches incurred by the optimal job schedule. The second step employs the KTNS policy to determine the tooling decision for any given job schedule. The third step consists of finding a job schedule that may incur a fewer number of tool switches by perturbing the current best job schedule. The disadvantage here is that the best sequence found by the heuristic does not guarantee a reduction on the makespan since the models does not consider production variables (demand, batch size, run time, etc.). Also the model assumes that the tool magazine is full loaded at each instant. This might increase the manufacturing costs when dealing with more than one machine center where tools are shared among the machines.

Barnea and Sipper (1993) developed an heuristic approach to reduce setup time in PCBs assembly. It considers the scheduling problem for a single machine, where jobs are queued and the machine has limited component capacity. When switching from one

job to another some components have to be replaced. The objective is to minimize total makespan for the fixed set of jobs. The problem is viewed as two interconnected sub-problems, the sequence problem (finding the job production sequence) and the component mix problem (finding the job component mix needed in the feeder of the insertion machine). The method generates a partial sequence in each iteration applying on this sequence the KTNS (keep Tool Needed Soonest) rule (Tang, 1988). In the sequence algorithm the selection process is guided by the principle of minimum change in the total number of component replacements in the planning horizon while the production constraints are satisfied. This process is based upon a selection index, three different indices for the selection process were tried as follows:

- Similarity index: number of common components of the next job and the current status of the feeder, where the decision criterion is maximization of this index.
- Variability index: same as above for the number of different components, where the decision criterion is the minimization of this index.
- Ratio index: the ratio between the similarity index and the total number of the different components of the job, where the decision criterion is the maximization of this index.

The mix algorithm basically uses a version of the KTNS rule an its properties, suggested by Tang (1988). Each iteration of the sequence algorithm gives the mix algorithm new information about coming jobs (look-ahead) and thus enables updating the components mix of previous iterations. One of the results obtained from the series of experiments is the rejection of the hypothesis that the sequence has significant impact on the number of setups. In other words the dominant factor is the mix and not the sequence. This assumes a surplus of feeder capacity. This approach differs from that of Tang (1988) only in that the heuristic generates a complete sequence in each iteration, whereas the proposed algorithm uses an "add-on" procedure, where a new job is introduced into the sequence at each iteration. Therefore the disadvantages here are the same as the ones from the algorithm proposed by Tang.

Sadiq and Landers (1992) and Sadiq, Landers and Taylor (1993) developed the intelligent slot assignment algorithm (ISA). It is a knowledge-based approach designed to sequence PCBs for assembly on one machine so that total production time is minimized. The algorithm uses the following rules when making slot assignments or changing parts on the machines:

- Add new parts in any empty slot(s) available.
- Replace those parts that have a history of minimal usage first.
- Consider the PCBs that are going to be populated in the near future because it may not be a good choice to take a part off the machine that would be needed to populate one of the following PCB types in the production sequence.
- Assign the parts for a job adjacently in a contiguous group and/or assign the highest-usage parts to slot positions promoting placement speed.

The algorithm uses three types of databases. The first is the CAD database containing a pool of assembly-related data (e.g. components which populated a PCB, location coordinates on the PCB, height, width, etc.) The second is the slot-definition database which contains dynamic slot-assignment information (part number, slot number, number of slots required by that part, history of usage and job-usage). The history field is useful when replacement of a part on a machine is performed, by choosing the parts which have been used a small number of times in the past as the candidate to be removed. The job-usage field contains dynamic information about future usage of the components and is updated after each job. The component-definition database contains information about all the components used in the shop. It includes the part number, the number of adjacent slots the part requires on the machine, and the history of the part usage. Unlike the slot-definition database, which includes only those parts that are on the machine, the component-database contains all the parts used (or to be used) in the shop.

The algorithm receives as input the CAD databases for the jobs assigned to the machine and suggests the optimal or near-optimal sequence of jobs. When assigning part

feeders to slots, the algorithm considers each part's future and past usage as well as the fact that a large part, requiring two slots on a machine, cannot be loaded until two single-slot parts are first removed. Once a sequence is selected, reassignment of parts is considered to reduce the placement cycle time.

Reassignment is the installation of all parts used on the current job inside a contiguous group of slots on the machine so that all the parts are adjacent to one another. The algorithm compares the benefit of reduced run time versus the cost of setup time to relocate the parts in a group of slots. For each possible sequence generated by the reassignment, the algorithm divides the number of saved moves by 27 (the authors assumed that the setup time for a part change is approximately equal to the run time of 27 carrier moves) to determine the allowable number of part changes to break even between setup and run time. If the allowable number of part changes exceeds the required number of part changes, then reassignment is economically feasible.

The major difference between assigning parts on the machine and reassignment of parts is that the assignment procedure places new parts on the machine while the reassignment procedure rearranges the parts inside a group so as to make the parts for a job adjacent, thus saving feeder carrier movements. The drawback of this procedure is that it does not consider that at every new feeder carrier configuration the CNC machine program has to be changed. Consequently making this technique of no use on real environments since the time and costs associated with this reprogramming in a high mix low volume manufacturing environment would be disastrous.

Peters and Subramanian (1996) proposed a strategy for the operational planning problem in a flexible flow line for electronic assembly systems. The line consists of a series of production stages, each of which having one or more identical machines operating in parallel. The model considers the analysis of a single stage of the line consisting of multiple, identical placement machines and attempts to determined the balance between processing time and changeover time during system operation. The four primary issues are determining the assignment of products to machines, the sequence of

products on each machine, the assignment of components to feeder locations for each product and the component placement sequence for each product. The problem is formulated as an IP problem where the objective function is to minimize makespan ensuring that the capacity of the machines is not exceeded, that the components required for a particular product be present on the machine before processing of the product begin, and that no two products are scheduled on the same machine at the same time. Four strategies for solving the operational planning problem are proposed:

- Unique setup strategy: determines the machine state and placement sequence assuming an "empty" machine (no feeders staged on the machine).
- Sequence dependent setup: determines the allocation of products to machines and sequence of products on each machine to minimize the makespan. Since a fixed machine state and placement sequence are used, the processing time is fixed and so the product allocation and sequencing decisions only affect the changeover time.
- Minimum setup strategy: chooses the machine state for processing a determined product to minimize the changeover from the current machine state. That is, only perform the required setup for product i on machine k.
- Tradeoff setup strategy: chooses specific optional setups that balance the tradeoff between processing time and changeover time. This strategy is similar to the method suggested by Sadiq (1993).

In the unique setup strategy, an initial component to feeder assignment is determined using a centroid allocation rule. The component type having the highest frequency of occurrence on the product is chosen and loaded in an empty feeder slot nearest to the centroid of that component type, where the centroid is the centroid location of all placement locations for that component type on the board. Given this component to feeder assignment, the component placement sequence is modeled as a traveling salesman problem, with the nodes in the network being the placement locations on the PCB and the arcs representing the time required to move from the previous placement location to the feeder containing the component required for the placement to the next placement location on the PCB. This subproblem is modeled as a Quadratic Assignment

Problem and a simple nearest neighbor heuristic is used to solve it. The products are then allocated to machines in order to minimize makespan by the use of a Multi-Salesmen Traveling Salesmen Problem (MTSP).

The sequence dependent strategy uses the component to feeder assignment and the placement sequence specified by the unique setup strategy in order to determine the processing time for each product and the sequence dependent changeover time for configuring the machine for product j given that it is currently configured for product i. The MTSP is used to determine the allocation of products to machines and the sequence of each product on each machine to minimize makespan.

The minimum setup strategy determines the allocation of products to machines and the sequence of products on each machine. It tries to minimize the changeover time as well as keep an even workload balance among machines. In the implementation the feeders that have the least frequency of use on the remaining products not yet allocated to a machine are removed, and components are assigned in order of frequency of use on the current product, to feeder slots using the centroid rule.

The tradeoff setup strategy begins with the minimum setup solution and systematically proceeds to modify the component to feeder assignment for each product, thereby reducing the placement time, aiming at improving makespan. It is achieved by moving the feeder having the highest frequency of occurrence to the position it would occupy if a unique setup were used for the product. It is a greedy algorithm that rearranges the feeders for a product if it minimizes the contribution of the product to the makespan, hoping that in the process the overall makespan will be minimized. Disregarding the fact that rearranging feeders for every product is impractical, one of the disadvantages of the proposed approach is that it assumes that a component type can be presented on more than one machine. This can substantially increase inventory costs. Also it assumes that a component occupies only one feeder slot, which in real situations might reduce the machine utilization since large components have to be placed in other machines. Moreover the model considers only one machine in the flow line. Therefore

the solution achieved will reduce the makespan only in that stage and not for the whole line. Finally the assumption of identical machines over simplify the problem making it a bit unrealistic.

Luzzato and Perona (1993) developed a procedure for grouping PCBs in a number of cells with minimized setup and maximized workload balance watching for satisfying the quantitative constraints as the number of available inserter machines at each assembly stage, the capacity of each available machine and the maximum number of component types that can be simultaneously equipped on each machine. The heuristic first assign boards and required components to Temporary Cells (TC). Three different strategies have been identified in order to choose the boards to be assigned:

- Strategy 1: The free PCB *i* with minimum NCA (number of component types added by PCB *i* to TC) is selected. If more than one PCB have the same NCA value, the one with the maximum NIA (number of insertions added by PCB *i* to TC) is selected among them.
- Strategy 2: The free PCB *i* with minimum NCA is selected. If more than one PCB have the same NCA value, the one with the minimum NIA is selected among them.
- Strategy 3: The free PCB selection policy is dynamically defined based on the current configuration of TC, with the objective to achieve a balanced exploitation of both the production capacity and the component types capacity of the cell. The strategies are chosen according to the following criteria:
  1. If $COM_s \geq CAP_s$ follow strategy 1.
  2. If $COM_s < CAP_s$ The free PCB *i* with maximum NCA is selected. If more than one PCB has the same NCA value, the one with the minimum NIA is selected.

Where CAP_s is the percentage of the objective utilization rate of an insertion machine of type s used by TC and COM_s is the percentage of the maximum number of component types per cell exploited by TC.

Every time the maximum number of component types that can be simultaneously equipped in the machine is reached or every time the maximum utilization rate (UR)

allowed is reached the TC is closed and a new cell is started. The temporary cells are then analyzed in terms of the utilization rates. The ones with minimum difference between the objective UR and the one obtained for that formation are selected and become definitive cells (DC). An attempted is made to insert free PCBs in the existing DCs through increasing the objective UR by a fixed user defined quantity so that less cells can be created. The drawback of this approach is that it considers grouping PCBs by the number of components in common. It allows a global setup minimization ensuring load balance but does not consider the sequence problem which is the most important cause of changeover between jobs.

Ben-Arieh and Dror (1990) studied the problem of assigning components to insertion machines with the aim of maximizing output. Two insertion machines are considered and the entire set of components types must be partitioned into two disjoint subsets which are loaded into the insertion machines. The problem is broken down into two cases:

- Case 1: each component type can be assigned to only one machine.
- Case 2: each component type can be assigned to both machines.

The objective function focus on balancing the total production load between machines by distributing components between them. It is formulated as follows:

$$\text{minimize} \sum_{i=1}^{N} C^i (X_{i1} - X_{i2})$$

where:

C^i = number of components of type i

X_{ij} = 1 if component type i is assigned to set N_j

0 otherwise

N = entire set of components to be inserted

j = insertion machines, j=1,2

It can be noticed that the model assumes that the insertion times for each component is the same, regardless of the circuit board type, component location on the board, component type, and insertion machines. This approach oversimplifies the problem making it very unrealistic. Therefore, although this technique may allow for a more global setup minimization, it does not ensure a sufficient load balance since the workloads of groups are not considered in the formulation. The drawback is the accumulation of work in process between the machines and maybe the totally infeasibility in terms of the technological constraints which are not included in the model. Also there is no practical justification for the assumption of case 2.

Cunningham and Browne (1986) developed a heuristic for job scheduling in printed circuit board assembly. The scheduling problem described is of a single machine (sequencer) problem with job dependent setup times and sequence dependent changeover times. The problem is defined as having two components:

- Decide on the order to schedule the jobs on the sequencer to minimize the number of reel changes, and
- Discover the "best" succession of reel changes for this schedule.

The problem of scheduling jobs with sequence dependent setup is NP-complete. Therefore the heuristic first divide jobs into groups containing similar jobs. This yields an acceptable level of performance with a loss of optimality in the schedules found: the schedules found are only optimal within groups. However, the overall schedule will still be near-optimal. For the problem of sequencing one job after the other the important criteria is how many components on the second reel are not on the first. Therefore two reels are grouped together if there are few different elements between them. The grouping procedure used first divide objects (reels) into groups at random and then takes the worst object in each group and places it in another group where it fits better. At the end it takes each object in each group and places it in another group if it fits better. This is repeated until no improvements can be made. The appropriateness of each reel to its group is assessed using a coefficient of similarity. A matrix indicating the number of components that have to be changed for every sequence is constructed as follows:

	A1	A2	A3	A4	A5
A1	0	7	3	0	4
A2	2	0	7	3	2
A3	5	3	0	0	13
A4	7	11	13	0	11
A5	13	15	2	1	0

The aggregate distance from A1 to all the other elements in its group is:

$$(7+3+0+4+2+5+7+13)/5 = 41/5$$

This value is used for choosing the worst element in a group. In order to control the size of the resulting groups large groups are broken up into smaller groups (the maximum size is determined by the user). The groups are sequenced using a branch and bound consisting of two stages. In the first stage a list of partial sequences is under consideration. At each iteration the most promising partial sequence is extended one step until a complete sequence is found. then the remaining incomplete sequences are extended until they become more "costly" than the currently favored sequence. The drawback of the heuristic proposed is that it does not consider neither the number of components of each type nor the run time in the evaluation of the coefficient of similarity. A weighted average should be used in the matrix so as to give more accuracy to the program.

PROBLEM STATEMENT

Modern PCB assembly often consists of a large-variety and small-lot size manufacturing system which uses computerized machines to automatically place electronic components onto boards. In this system, setup reduction is the major concern in terms of productivity since these machines have rapid assembly rates but long setups. A typical machine is capable of assembling thousands of components per hour while its setup time may take about an hour.

The machines are arranged in a flowshop type assembly line. Frequently, lines are arranged according to technological and production rate capabilities. Products are assigned to lines according to technological and demand requirements. There is no concern of assembly costs or setup reduction when assigning boards to lines.

Typically each board requires a variety of component types. Components are placed in feeders which are fixed on the machine's feeder carrier. The carrier has a limited capacity and cannot accommodate component feeders for all jobs scheduled in the machine. Therefore, at the end of each job, a setup of the component feeders necessary for the next job is incurred. Thus, the number of feeders to be changed and, consequently the setup time, depends on the number of component types required by the new board that are currently not available on the machine. Hence, it is preferred to sequence jobs so as to chose those that have more common parts with the one currently on the machine.

Therefore, the approach proposed here focuses on grouping PCBs according to their similarity and assessment of possible sequences attempting to develop the assignment of boards to lines. It focuses on reducing the manufacturing costs by grouping boards and lines according to the demands and machine capabilities and by increasing the productivity through the reduction of changeovers. It assumes that a real improvement can be achieved if boards are grouped together in families so as to reduce setup times within the families. Since this setup is sequence dependent, boards have to be

grouped according to their similarity and attempting to achieve an overall changeover reduction in a suggested sequence.

Basic Assumptions

- Since the objective is to reduce the total number of setups, lines are treated as if they were a large assembly machine.
- There is an order constraint on the PCBs assembly which forces them to be processed on a flowshop type assembly line. The reason is that larger components (integrated circuits - ICs/ dual-in-line packages - DIPs) should be assembled after the smaller ones (chip, resistors, capacitors, SOIC's) because the machine's head can hit the larger components if they are inserted first.
- The setup time considered is only the setup time required when the product type to be assembled is changed. Replenishing components in the machines during the assembly of a lot of identical PCBs is not considered because the amount of components required for the assembly of each PCB does not depend on the production method used. Also, in some cases, the machine can continue assembling during the refilling operation, while during the setup for a new product, it must be idle.
- There is no concern with the routing of the machine's head on the PCB while assembling the components. The routing problem is a separate problem, dealt with extensively in the literature.
- The boards (flats) are assumed to have the same dimensions and therefore no setup is incurred when changing from one board to other (only component feeders are changed).
- Component feeders are assumed to have the same characteristics and therefore their setup time is the same.
- For the environment studied, the processing time of each PCB in the system is largely a function of the number of components of each type mounted. Component

feeders are loaded onto each placement machine before starting production of a board. Machines have limited space for holding feeders.

Goals and Objectives

In summary, the main objective of the study presented here is to design and test a procedure capable of assigning boards to lines with minimized setup which satisfies the quantitative constraints, such as product demand and capacity of each available line, and which is concerned to sequence dependency. The goal is to propose a heuristic for assigning PCBs of a given mix to placement machines in such a way that if the boards in a group are scheduled sequentially, setups and manufacturing costs are minimized, and machine utilization is maximized.

MODEL DEVELOPMENT

Boards and parts are assigned to PCB assembly lines with the objective of attending the required demand and restricted by technological and process constraints. After the assignment, High Speed Placement Machines (HSPM) CNC software attempts to optimize the travel distance of the head when placing components as well as optimizing the travel distance of the component feeder carrier. Although the position of components in the carrier can be fixed and programs can be created accordingly, usually programs are done individually for each board type, where the software determines different feeder positions in the carrier for each board. Therefore when the PCB batch is changed, some of the component feeders must be re-setup. In high mix low volume processes this setup accounts for most of the manufacturing costs since the equipment remains idle during setup.

The primary way of dealing with this problem is to assign boards with the maximum number of components in common to the same line. It does not guarantee a serious reduction in setup since other variables are not considered. For example, the assignment to a line of two similar boards, one with high demand and other with very low demand, will not significantly improve the overall system since the one with high demand would be scheduled more often than the other. Therefore the difference in setup from this board to the other ones would be the factor which would determine the amount of setup for that line. Furthermore, the CNC software may place the same component in different positions in the carrier for each board, which will result in a setup if changing from one board to the other. This can be avoided by forcing the software to use a unique position for that component, but will result in an increase in the runtime, since the travelling distances of the feeder carrier will not be optimized.

Researchers in the past have concentrated on using only similarity as a mean of reducing/eliminating changeovers in PCB assembly. This approach is overly simplistic in the problem of assigning boards to lines attempting to an overall setup/costs reduction.

Other variables like demands, runtimes, costs and most of all sequence dependency should be considered. This research aims at a more realistic approach by simultaneously considering the effects of these other variables plus the possibility of assigning boards to other lines in an attempt to obtain a better solution by improving the component mix.

Notations used in the mathematical model followed by the development of the actual model are presented in the following sections.

Notations

$d_i =$ annual demand for board i

$X_{i,l} =$ 1 if part i is assigned line l

0 otherwise

$Y^l_{i,j,k} =$ 1 if part i is assigned to position j and part k to position (j - 1) in line l

0 otherwise

$S^l_i =$ setup costs for board i in line l (per board)

$RC_{i,l} =$ run time costs for board i in line l

$at_l =$ available capacity on line l

$\tau^l_{S_{i,k}} =$ setup time incurred if part k is followed by part i on line l

$\tau_{R_{i,l}} =$ run time of board i on line l

$b_i =$ batch size part i

$\lceil \bullet \rceil =$ smallest integer that is greater than or equal to "$\bullet$"

$F_l =$ performance factor of line l

$k = i = 1,2,\ldots, N$ boards

$l=$ 1,2,..., L lines

$j=$ 1,2,..., Z positions

Model Description

A general binary integer linear programming model is formulated for the problem. Its objective is to minimize the total cost in a fixed time horizon. The objective function can be divided in two elements: runtime costs and setup costs. The optimization is done by finding the assignment of a group of boards to a set of lines which will result simultaneously in the minimum number of changeovers and in the minimum runtimes.

The minimum setup is determined by evaluating all possible sequences for the set of boards assigned to a line. Therefore when assigned to a line, boards are placed in several positions in the sequence. The number of changeovers and the total setup time incurred from changing from the previous board in the sequence to the current board is determined. After all boards are assigned to the set of lines, the total costs (runtime plus setup) is computed. Two binary constraints are applied to provide all possible combinations on the assignment to lines and on the sequencing of boards in a line. The result is the assignment of boards to lines and the sequence in which the boards should be scheduled.

The whole process is done over four feasibility constraints. First, the capacity constraint restricts the number of boards in a line so as to not exceed the available capacity in that line. A performance factor is used to assure that the solution found fits with the available resources. The second constraint assures that a board is assigned to only one line. The third constraint assures that a board is assigned to only one position in the sequence. The fourth constraint assures that runtime costs are computed only when a board is assigned to a position in a line sequence.

Mathematical Model

Minimize:

$$\sum_{l=1}^{L}\sum_{i=1}^{N} d_i.\tau_{R_{i,l}}.RC_{i,l}.X_{i,l} + \sum_{l=1}^{L}\sum_{j=1}^{Z}\sum_{i=1}^{N}\sum_{k=1|k\neq i}^{N} S_i^l.\tau_{S_{i,k}}^l \left\lceil \frac{d_i}{b_i} \right\rceil .Y_{i,j,k}^l$$

subject to:

$$\sum_{l=1}^{L}\sum_{i=1}^{N} d_i.\tau_{R_{i,l}}.X_{i,l} + \sum_{l=1}^{L}\sum_{j=1}^{Z}\sum_{i=1}^{N}\sum_{k=1|k\neq i}^{N} \tau_{S_{i,k}}^l \left\lceil \frac{d_i}{b_i} \right\rceil .Y_{i,j,k}^l \leq \sum_{l=1}^{L} at_l.F_l$$

$$\sum_{l=0}^{L} X_{i,l} = 1 \qquad \text{for i = 1,2,...,N}$$

$$\sum_{l=1}^{L}\sum_{j=1}^{Z}\sum_{k=1|k\neq i}^{N} Y_{i,j,k}^l = 1 \qquad \text{for i = 1,2,...,N}$$

$$X_{i,l} = \begin{cases} 1 & \text{IF} \quad Y_{i,j,k}^l > 0 \\ 0 & \phantom{\text{IF}} \quad Y_{i,j,k}^l = 0 \end{cases} \qquad \text{for all i, j, k, and l}$$

Computational Complexity of the Problem

The mathematical model developed here is a binary integer linear programming problem. Most integer programming problems fall into the class of NP-complete problems (Garey and Johnson 1979). The algorithm can be divided in two sets. First, the line assignment. This algorithm is called an exponential time algorithm (Garey and

Johnson 1979) which time complexity function can be defined as w^n where w is the number of lines in the system and n is the number of boards to be assigned to lines. Second, the board sequencing problem. This is also an exponential time algorithm which has been compared and investigated as a Traveling Salesman problem (Hashiba and Chang 1992) and is recognized as an NP-complete problem. A similar problem involving sequencing a set of tasks and setup times and studied by Bruno and Downey (1978) has also proved that the sequencing problem is NP-complete in the strong sense.

The theory of NP-completeness proves that a given problem is "just as hard" as another problem which is recognized as being NP-complete (Garey and Johnson 1979). Therefore, since the assignment to lines with posterior sequencing can be viewed as a combination of these two NP-complete problems we can assure that it is at least as hard as the two portions. Thus, a higher level heuristic based upon a concept known as Genetic Algorithms is used to solve the problem.

Heuristic Algorithm

Introduction

Typically our goal in developing algorithms for combinatorial optimization is to find good solutions relatively quickly. The approach presented here introduces a two layered strategy. At the top or global level we partition the search space into promising regions using a genetic algorithm. The second level then conducts a detailed search within the promising partitions.

The genetic algorithm (GA) procedure used to solve the PCB-assignment problem is a nontraditional methodology that uses a real-valued representation and evolutionary procedure as developed by Michaelwicz (1992). The procedure is summarized in the following sections.

Genetic Algorithms

Genetic algorithms mimic the evolutionary process through the use of a "survival of the fittest" strategy. In general, the fittest individuals of any population tend to reproduce and pass their genes on to the next generation, thus improving successive generations. However, some of the worst individuals might survive and also reproduce. Genetic algorithms have been shown to solve linear and nonlinear problems by exploring regions of the state space and exponentially exploiting promising areas through mutation, crossover, and selection operations applied to individuals in the population (Michaelwicz, 1992).

Whereas traditional search techniques use characteristics of the problem to determine the next sampling point (e.g. gradients, Hessians, linearity and continuity), Genetic Algorithms (as well as all stochastic search techniques) make no such

assumptions. Instead, the next sampled point(s) is (are) determined based on stochastic sampling/decision rules rather than a set of deterministic decision rules.

Genetic algorithms generate and maintain a family, or population, of solutions by using a selection mechanism. This provides the exploitation of several promising areas of the solution space at the same time. Solutions are encoded as sequences of digits from a binary alphabet. As in nature, selection provides the necessary driving mechanism for better solutions to survive. Each solution is associated with a fitness value that reflects how good it is, compared with other solutions in the population. The higher the fitness value of a member of the population, the higher its chances of survival and reproduction and the larger its representation in the subsequent generation.

Recombination of genetic material in genetic algorithms is simulated through crossover, a mechanism of probabilistic and useful exchange of information among solutions that exchanges portions between sequences of digits. Another operation, called mutation, causes sporadic and random alteration of the digits on a sequence.

Encoding mechanism

Fundamental to the GA structure is the encoding mechanism for representing the optimization problem's variables. Each variable is first linearly mapped to an integer defined in a specified range, and the integer is encoded using a fixed number of binary bits.

In the PCB - line assignment problem, the variables dealt with are the lines which must be assigned boards. Therefore, in order to obtain a sequence of digits, lines are represented by integer numbers ranging from zero to the maximum number of lines minus one. The digits in the string represent the boards which have to be assigned to the lines.

For example, consider a problem where ten boards (0,1,...,9) have to be assigned to three lines (0,1,2). The GA could create an assignment as follows:

2,2,0,1,0,2,1,0,0,1

Every board is represented by the corresponding digit in the increasing order (from left to right) and every line is represented by an integer (0 for line 0, 1 for line 1, and so on). Therefore, this representation would send boards 0,1, and 5 to line 2. Boards 2,4,7, and 8 to line 0, and boards 3,6, and 9 to line 1.

Fitness function

The objective function, the function to be optimized, provides the mechanism for evaluating each string. However, its range of values varies from problem to problem. To maintain uniformity over various problem domains, we use the *fitness function* to normalize the objective functions to a convenient range of 0 to 1. The normalized value of the objective function is the fitness of the solution.

In the PCB - line assignment problem, after a population is formed and the objective function value (total manufacturing cost of that assignment) of every member in the population is determined, the sum of deviation values of every member to the worst member is determined as follows:

$$dev = \sum_{i=0}^{N} Z_{\max} - Z_i$$

where:

dev = total deviation

Z = objective function value

$Z_{\max}$ = maximum objective function value found in the population

i = population member (0,1,..., N)

The fitness value is then determined as the contribution in the deviation of every member in the population. This is calculated as follows:

$$fitness(i) = \frac{Z_{\max} - Z_i}{dev}$$

It can be seen that the member with the minimum objective function value will have the highest fitness value, therefore with highest probability of being reproduced in the following generations and the highest probability of being selected for the following operations. A cumulative fitness (*cumfitness*) value is also determined and given to every member by adding up all the fitness values of the previous members in the population:

$$cumfitness(i) = \sum_{k=0}^{k=i} fitness(k)$$

This value will be used for the *roulette wheel selection scheme* described in the following section. The flowchart of the fitness determination function is shown in Figure 5 where *popsize* is the population size and cost(*i*) and *max.cost* are the objective function values of member i and of the worse member in the population respectively. Note that if the deviation is smaller than a user defined value (the solutions are not meaningfully different) a fixed fitness value (1/popsize) is assigned to all members of the population.

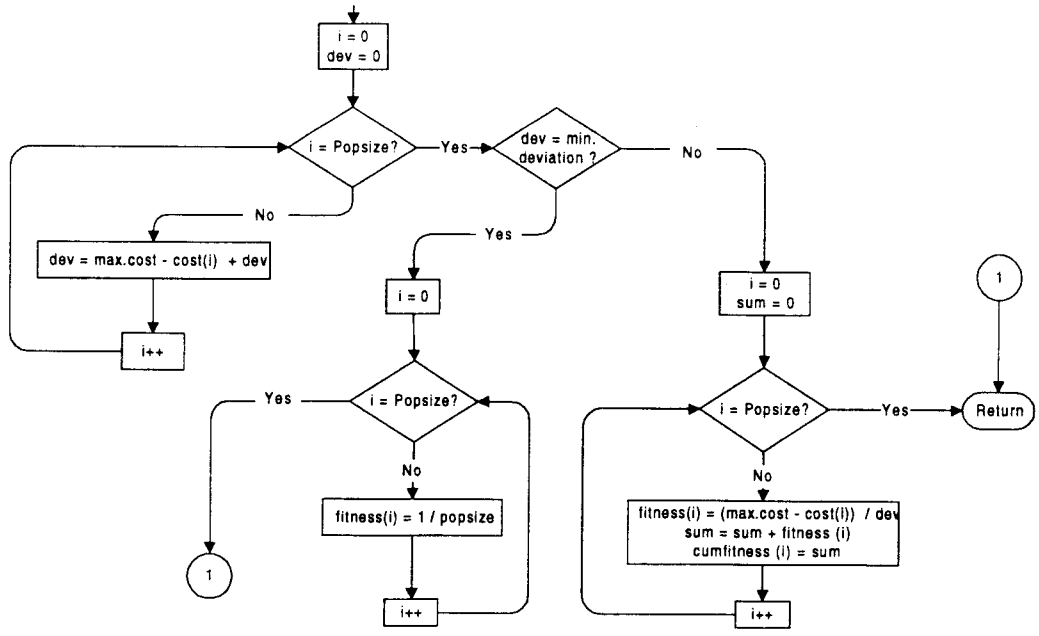


Figure 5 - Flowchart of the fitness determination

Parent Selection

Selection models nature's survival-of-the-fittest mechanism. The purpose is to give more reproductive chances, on the whole, to those population members that are most fit. Fitter solutions survive while weaker ones perish. The PCB - line assignment problem uses a commonly-used technique: the Roulette Wheel Selection. Its flowchart is shown in Figure 6 which can be divided into three steps:

- Determine the cumulative fitness of all the population members
- Generate a random number (p) between 0 and 1
- Select population member according to the following criteria:

  $cumfitness(j) < p < cumfitness(j + 1)$ select population member (j + 1)

  $cumfitness(0) > p$ select population member (0)

This algorithm is referred to as roulette wheel selection because it can be viewed as allocating pie-shaped slices on a roulette wheel to population member, with each slice proportional to the member's fitness (Davis 1991). Selection of a population member to be a parent can then be viewed as a spin of the wheel, with the winning population member being the one in whose slice the roulette spinner ends up. This technique has the advantage that it directly promotes reproduction of the fittest population members by biasing each member's chances of selection in accord with its evaluation.

The effect of roulette wheel selection is to return a randomly selected parent. Although this procedure is random, each parent's chance of being selected is directly proportional to its fitness. On balance, over a number of generations this algorithm will drive out the least fit members and contribute to the spread of the genetic material in the fittest population members. Of course, it is possible that the worst population member could be selected by this algorithm each time it is used. Such an occurrence would inhibit the performance of the GA, but the odds of this happening in a population of any size is negligible.

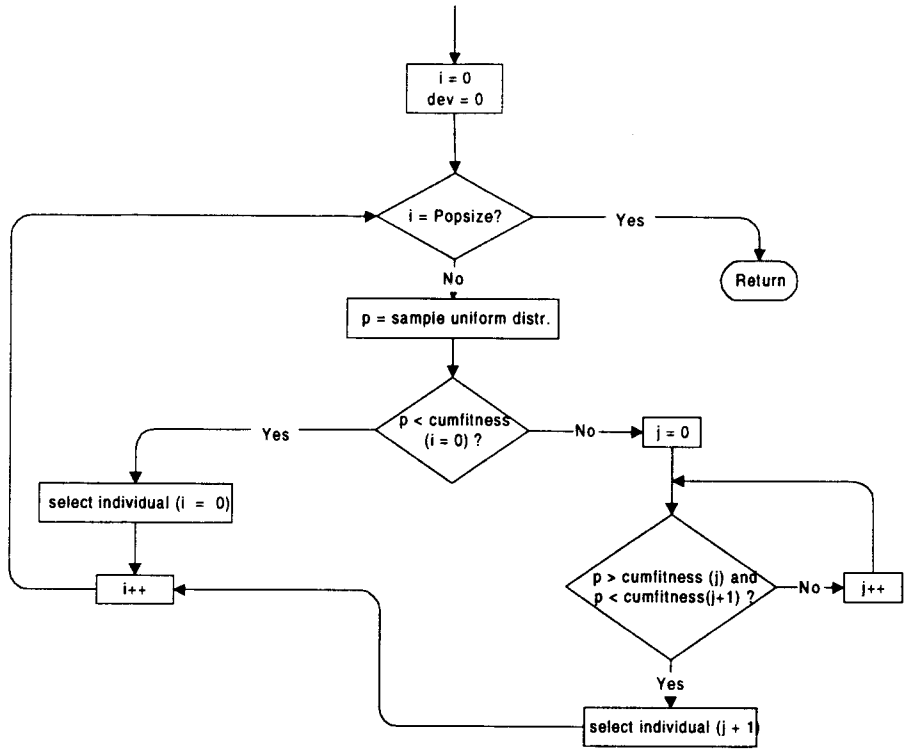


Figure 6 - Flowchart of the selection operation

This parent selection algorithm is used in many stages throughout this implementation. It is applied in the selection of a new generation, as well as in the selection of members for crossover and mutation.

Crossover

After selection comes crossover, GA's crucial operation. Pairs of strings are picked at random from the population to be subjected to crossover. The single-point crossover (Holland 1975) is applied in the PCB-line assignment problem. Here, parts of two parent chromosomes are swapped after a randomly selected point, creating two children. Following an example of the application of one-point crossover during a GA run, the children are made by cutting the parents at the point denoted by the vertical line and exchanging parental genetic material after the cut:

Parent 1: 1 1 1 1 | 1 1		Child 1: 1 1 1 1 0 0
	⇒	
Parent 2: 0 0 0 0 | 0 0		Child 2: 0 0 0 0 1 1

One important feature of one-point crossover is that it can produce children that are radically different from their parents. Another feature is that it will not introduce differences for a digit in a position where both parents have the same value allowing to keep a possible schema. A schema is a similarity template describing a subset of strings with similarities at certain positions (Goldberg 1989). In other words, a schema represents a subset of all possible strings that have the same digits at certain string positions. For example, a schema **000 represents strings with 0s in the last three positions: the set of strings 00000,01000,10000, etc.

The Schema Theorem (Goldberg 1989) says that a schema occurring in chromosomes with above-average evaluations will tend to occur more frequently in the next generation and vice versa. GA manipulate a large number of schema in parallel. The reproduction mechanisms together with crossover cause the best schemata to proliferate in the population, combining and recombining to produce high-quality combinations of schemata on single chromosomes.

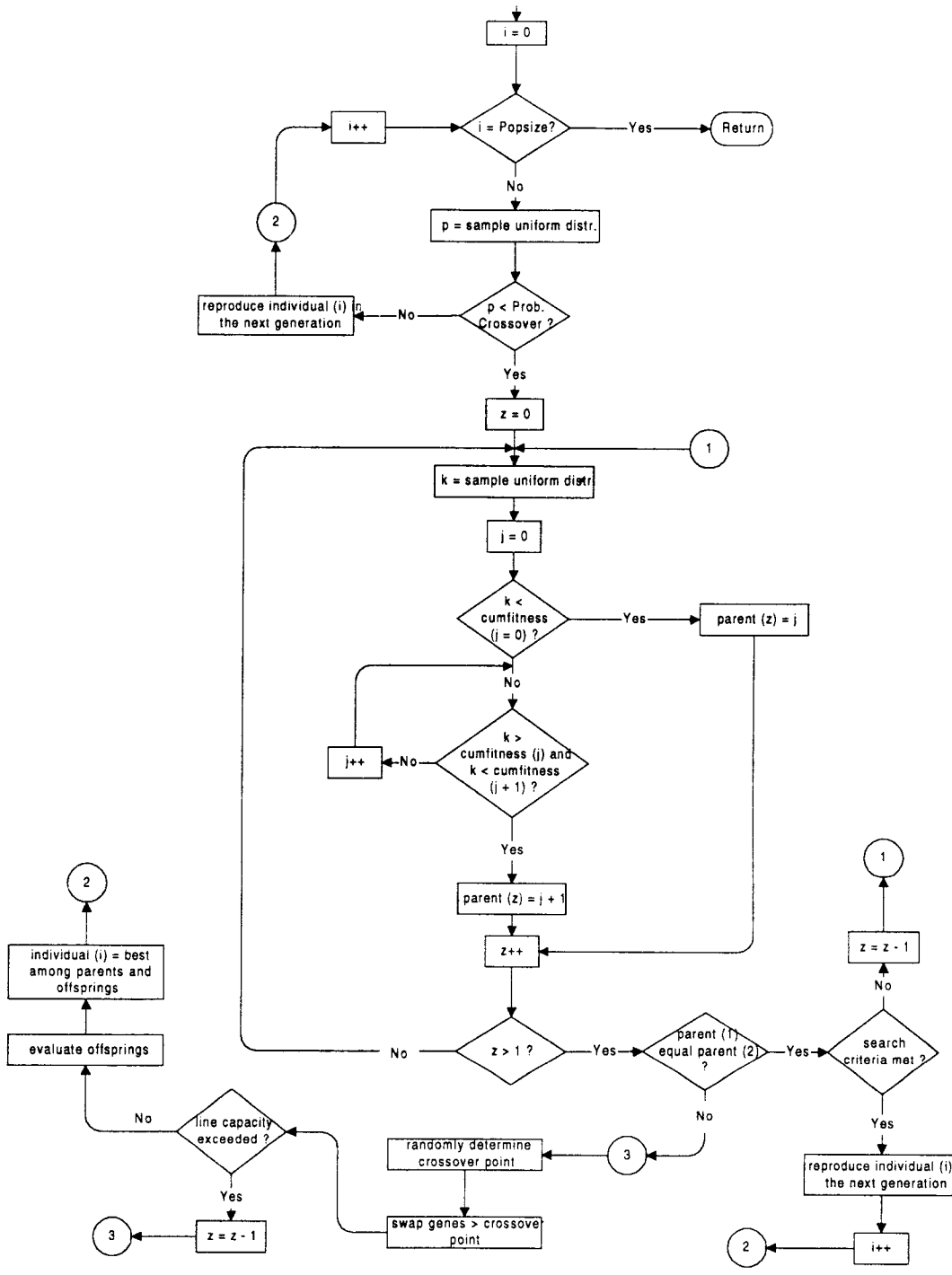


Figure 7 - Flowchart of the crossover operation

Figure 7 shows the flowchart of the crossover function used in the PCB-line assignment problem. Initially, a population member of the new generation is chosen to

be created by crossover if a randomly generated number in the range 0 to 1 is greater than or equal to a user defined probability of crossover (*Pc*). Ultimately, the value of *Pc* in a large population gives the fraction of strings actually crossed. Then, two parents are selected by applying the parent selection algorithm described previously. An attempt to find two different parents is made (since children of the same parents are equal to the parents). If a search criteria (the number of tries to find a different parent) is reached the member of the old population is simply reproduced in the new generation. If not, assuming that L is the string length, it randomly chooses a crossover point that can assume values in the range 1 to L - 1. The portions of the two strings beyond this crossover point are exchanged to form two new strings. The crossover point may assume any of the L - 1 possible values with equal probability. Both children are checked for feasibility and then evaluated. The member (child or parent) with the lowest objective function value is reproduced in the new generation.

Mutation

After crossover, strings are subjected to mutation. Mutation of a digit involves flipping it: changing a 0 to 1 or 2, or vice versa. The probability of mutation (*Pm*) gives the probability that a member will be mutated. The digits of a string are independently mutated, that is, the mutation of a digit does not affect the probability of mutation of other digits. Mutation is dealt only as a secondary operator with the role of restoring lost genetic material. For example, suppose all the strings in a population have converged to a 0 at a given position and the optimal solution has a 1 at that position. Then crossover cannot regenerate a 1 at that position, while a mutation could.

Figure 8 shows the flowchart of the mutation function. Initially, a population member of the new generation is chosen to be mutated (after crossover or reproduction) if a randomly generated number in the range 0 to 1 is less than or equal to a user defined probability of mutation (*Pm*). Assuming that L is the string length, a random number that can assume values in the range 0 to L - 1 is used to determine the digit to be mutated.

The digit is then flipped to a randomly determined number different than the digit number. For example, in case of three lines (0,1,2) if the digit chosen has the number 1 it can be changed to 0 or 2, this decision is made randomly. After the user determined number of digits to be mutated is reached, a feasibility test is made and the child is evaluated. The member (child or parent) with the lowest objective function value is reproduced in the new generation.

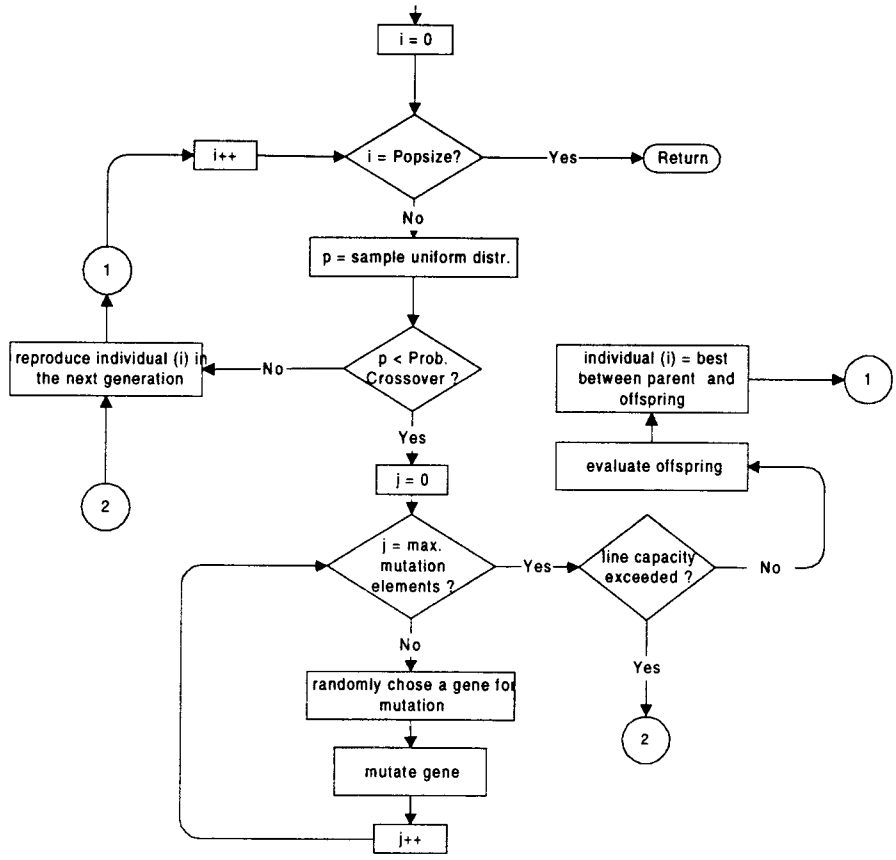


Figure 8 - Flowchart of the mutation operation

Generation cycle

Figure 9 shows a generation cycle of the GA with a population of four strings with 10 digits each. In this example the objective function values are randomly created. The fitness and cumulative fitness values are determined for each member. The selection rule allocates members to population P2 according to a uniformly distributed random number. Next, the four strings are paired randomly for crossover. Two members of the new population are randomly selected for crossover in the same manner. The other pair of strings is left intact. The crossover point is randomly determined and falls between the third and forth digits of the strings, and portions of strings 1 and 2 beyond the third digit are swapped. Population P3 represents the set of strings after crossover. The action of mutation on population P3 can be seen in population P4 on the third and seventh digits of string 2 and the first and seventh digit of string 4. Population P4 represents the next generation. Effectively, P1 and P4 are the populations, while P2 and P3 represent intermediate stages in the generation cycle.

Population P1:

String	Objective Function Value	Fitness	Cumulative Fitness
0000001111	7951	0.209	0.209
1000120120	9070	0.000	0.209
2200101110	6277	0.523	0.732
1100012222	7640	0.268	1.000

Population P2: After Selection

String	Objective Function Value	Fitness	Random Number
1100012222	7640	0.268	0.833
2200101110	6277	0.523	0.432
2200101110	6277	0.523	0.636
1100012222	7640	0.268	0.914

Population P3: After Crossover

String	Objective Function Value	Fitness	Cumulative Fitness
110|0101110	7370	0.275	0.275
220|0012222	8669	0.000	0.275
2200101110	6277	0.507	0.782
1100012222	7640	0.218	1.000

Population P4: After Mutation

String	Objective Function Value	Fitness	Cumulative Fitness
1100101110	7370	0.133	0.133
2210010222	7244	0.195	0.328
2200101110	6277	0.672	1.000
2100010222	7640	0.000	1.000

Figure 9 - Generation Cycle

Objective Function Evaluation

One of the major problems of the assignment of boards to lines is the number of setups produced by that assignment. Since placement machines have fast placement rates and therefore low runtimes, changeovers count for a large share of equipment available time. Consequently, the only way to obtain a realistic evaluation of that assignment is by including costs concerning setup times. Although runtime costs can be easily determined, setup costs can only be evaluated by sequencing the boards on each line.

Therefore, the problem is divided into two steps: the assignment of boards to lines by the GA and a local optimization algorithm. By this way, the GA is used to explore new regions in the search space and to exploit already sampled regions. The local optimization algorithm is performed in order to find a local optimum solution resulting in less setup. A multistart algorithm applying local search is used to each of a large number of random initial feasible points selecting the best of the local optima.

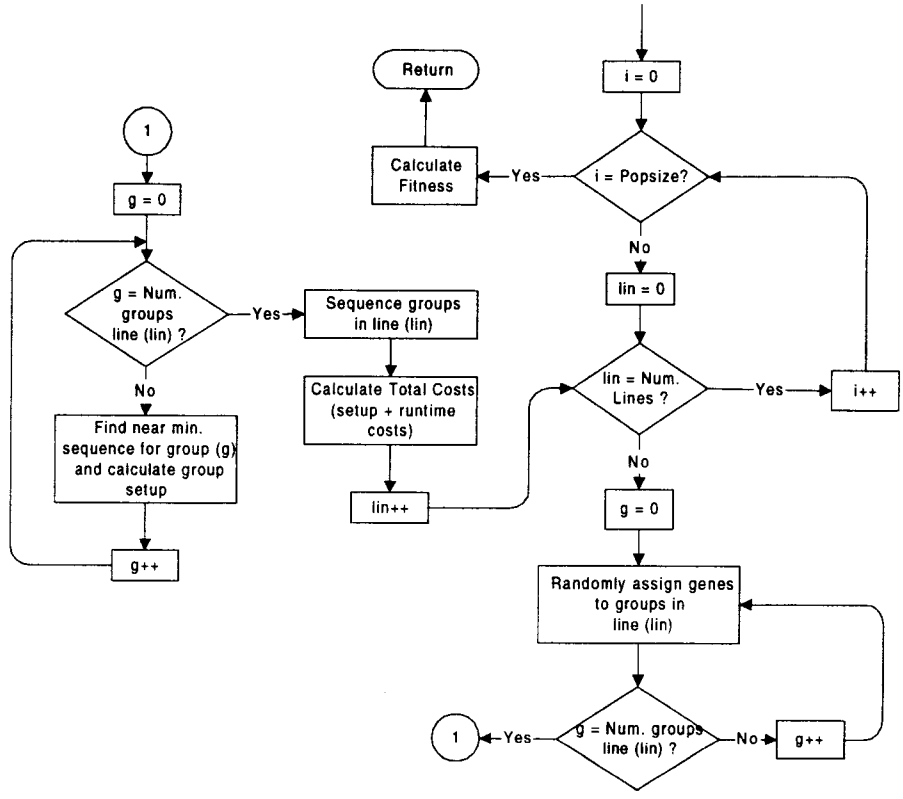


Figure 10 - Flowchart of the population members evaluation

Figure 10 shows the flowchart of the function used to evaluate the objective function value. Initially, after the allocation to specific lines by the GA, boards are randomly assigned to groups. The number of groups will depend on the number of boards assigned to a line and on a user defined maximum group size. The assignment of boards to a group is accomplished by randomly choosing genes in the string. After all the

groups are formed in a line, the multistart algorithm is used to determine the local best sequence by generating random starting points and calculating setups by using the steepest descent path. Therefore, after the start, the branch resulting in minimum setup is taken consecutively for every board in the group. After finding the local optimal sequence for all groups the same multistart algorithm is used to determine the best sequence of groups. The total number of setups is then determined and the total cost of that assignment is calculated by adding the setup costs (TSC) and runtime costs (TRC) which are calculated as follows:

$$TSC = \sum_{l=1}^{L} NS_l * ST_l * LC_l$$

$$TRC = \sum_{l=1}^{L} \sum_{i=1}^{G} d_i . \tau_{R_{i,l}} . X_{i,l} * RC_L$$

where:

l = lines 1,2,..,L

G = Set of boards assigned to line l

d_i = annual demand for board i

$X_{i,l}$ = 1 if board i is assigned line l

0 otherwise

LC_l = line l costs

RC_l = run time costs for line l

NS_l = total number of setups in line l

ST_l = changeover time (basically the time of changing one feeder)

$\tau_{R_{i,l}}$ = run time of board i on line l

This process is repeated for every population member and the value is used on the fitness determination. This method does not assure that the solution is a near-global

optimum (especially if the variance between the local optima is large). A systematic way to ensure finding a near-optimal solution would be the use of an enumerative method (like the branch and bound technique). The drawback and consequently the reason for using the multi-start technique is that, since the assignments are made randomly and since the PCB line assignment problem deals with a large number of boards, the use of an enumerative method would be impractical. The multistart approach approximates the objective by selecting from a set of local optima.

Algorithm Steps

Over the last decade, considerable research has focused on improving GA performance. Efficient implementations of different selection and crossover mechanisms have been proposed as improvements on the traditional approaches and significant innovations have been achieved by the use of adaptive techniques (Srinivas and Patnaik 1994). Although the Simple Genetic Algorithm or SGA (Holland 1975) has been commonly used over the years, its use in two layered problems has increased recently. Therefore, three heuristics with different behavior regarding the two layers were developed and implemented. Different than the previous work, which concentrated on the development of the GA specifically, they differ on the interaction of the two levels (GA and the local optimum search). Their different steps are described in the following sections. The algorithm steps for the three heuristics were coded using ANSI C and run on a Pentium / 133 MHz with 64 Mb RAM.

Heuristic 1

Figure 11 shows the flowchart of Heuristic 1. This heuristic accounts for the importance of the second level of the problem (local search of minimum setup). This is accomplished by the continuous re-evaluation of the population. As previously mentioned, every time the population member is evaluated, boards belonging to a line

are randomly grouped and the sequencing routine is performed within a group and between groups. Chances are that by continuously re-evaluating the population we achieve a better solution in the second level which can result in a overall better solution.

Initially, the manufacturing data (demands, runtimes, line costs, setups from one board to the others, etc.) is randomly created. This was done to simplify the handling of problems of different sizes. It also contributes to eliminating any possible biases which might exist. The first step in any GA heuristic is the creation of the initial random population. This is performed by randomly assigning numbers representing the assembly lines to the string. For example, assuming there are three assembly lines, numbers between 0 and 2 will be randomly assigned to the digits in the string. Care is taken to not allow the creation of two population members with the same configuration (identical strings). Figure 12 shows the flowchart of the creation of the initial population.

The evaluation of the population is performed as described previously, so after the assignment, boards belonging to a line are randomly assigned to groups of user defined size, the multistart algorithm is used to find a local minimum sequence within a group and between groups. After the objective function value is determined for the entire population, fitness is calculated and the best element within a population is copied in another data structure and kept for later comparison.

A termination criteria is set to end a GA run. There are three important pieces on this criteria: the minimum number of generations, the maximum number of consecutive bests, and the maximum number of generations. For example, assuming the minimum number of generations is 10, the maximum number of consecutive bests is 5 and the maximum number of generations is 50. The run can only be terminated if after 10 generations the same best was achieve 5 times or if the total number of generations reaches 50. The first two criteria assure that, when reaching a local optima where minimum or no improvement can be achieved, the run can be terminated and a new region in the solution space can be investigated. The second and third criteria are used to sense that, although a better solution was achieved, there is still a chance of improving it.

The first criteria is used to not allow the search to be trapped in a local optima by forcing the investigation of new points in the search space.

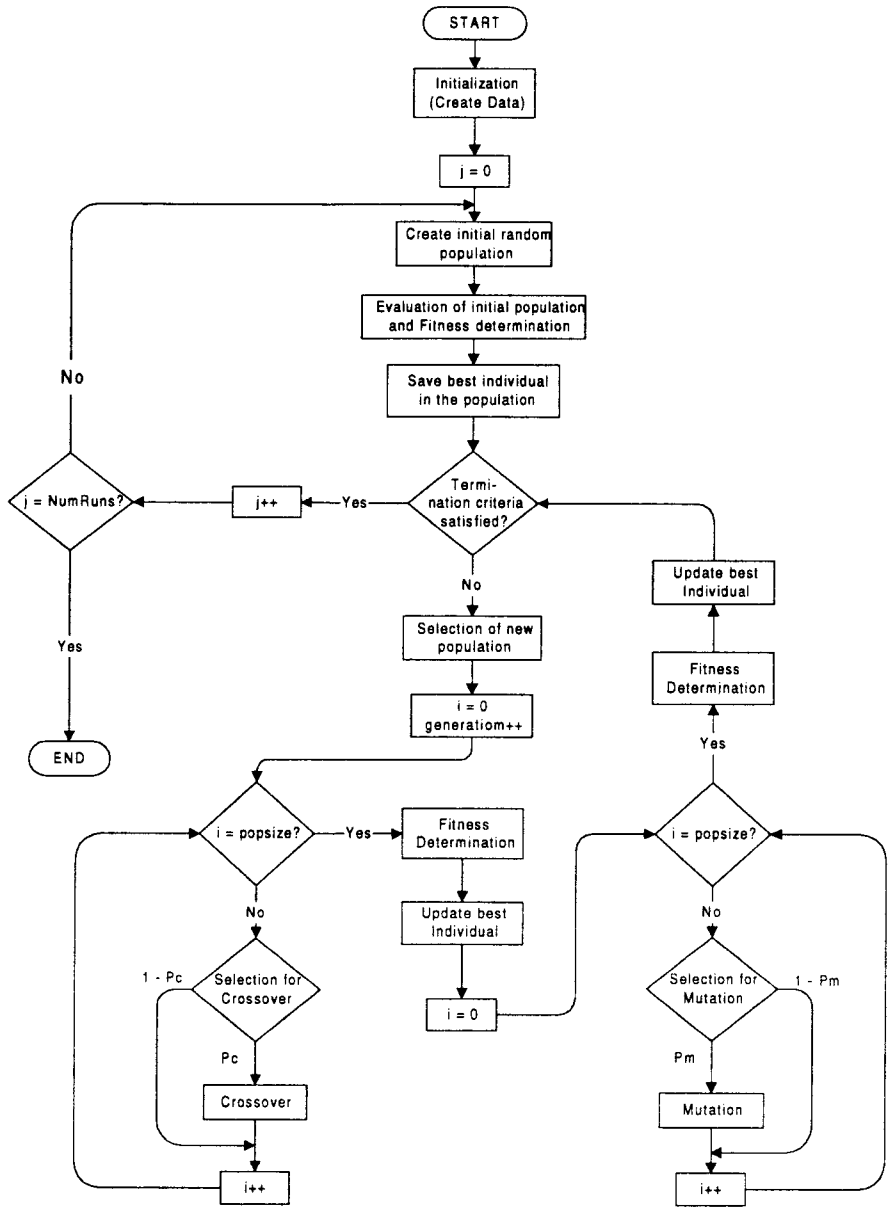


figure 11 - Flowchart Heuristic 1

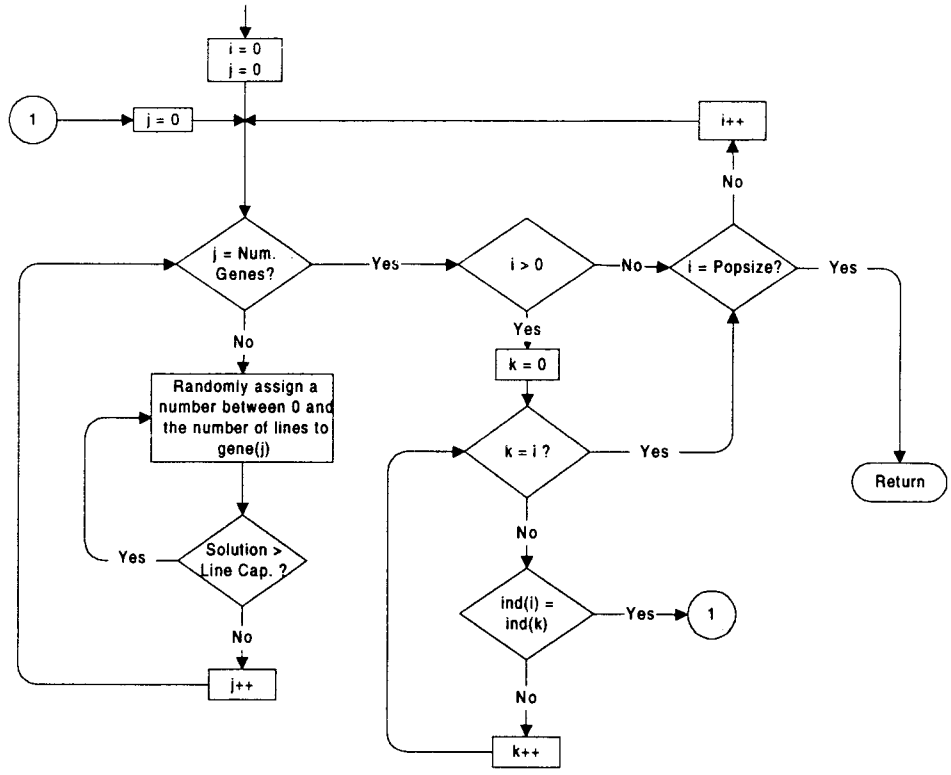


Figure 12 - Flowchart of the Creation of the Initial Population

Selection according to fitness is then performed as mentioned previously and a new population (or the intermediate stage in the generation cycle) is created. Then, the crossover operation is used on the creation of the new population. Therefore, a member of the new population is chosen to be created by crossover if a uniformly distributed random number in the range 0 to 1 is greater than or equal to a user defined probability of crossover (*Pc*). Two parents are chosen according to their fitness and the new member will be equal to the best member in this set (parents or children). Therefore the algorithm assures that the best is always saved. If the random number lies in the 1 - *Pc* interval, the member of the former population is just reproduced in the new one.

After the new population is created, fitness values are calculated and the best member in the new population is determined and compared to the former best. If the new solution is better, it is copied in the data structure.

A new population is then created by applying the mutation operation. As in the crossover operation, members of the new population are chosen to be created by mutation of members of the former population if a uniformly distributed random number in the range 0 to 1 is less than or equal to a user defined probability of mutation (*Pm*). The best between parent and child is reproduced in the new population. After the new population is created, fitness is determined and if the best member in the new population is better than the former best, it is copied in the data structure. The termination criteria is checked and if not reached a new generation is initiated following all the steps described previously. After the user defined maximum number of runs is reached the process is terminated and the best assignment is printed out as well as the set of groups per line and the sequence of boards in each group and the sequence of groups in each line.

Heuristic 2

Figure 13 shows the flowchart of Heuristic 2. This heuristic accounts for the importance of the first level of the problem (GA assignment of boards to lines). This is accomplished by performing the crossover operation on the creation of a new population followed by the mutation operation and only then re-evaluating the population regarding the second level (local search of minimum setup).

The justification for this approach is that in problems where the assignment accounts for the largest share in the objective function value (manufacturing costs) an improvement on the second level will not be as significant as an improvement in the first level. The crossover followed by mutation also allows the GA to focus in a larger fraction of the search space and reduces the probability of the crossover operation being trapped in a local optima. Very often, after a large fraction of the population has converged (the strings have become homogeneous), crossover becomes ineffective in searching for better strings. This approach also speeds the computation time, since the grouping and sequencing algorithms are performed only in the end of the GA cycle.

As in Heuristic 1, initially the manufacturing data (demands, runtimes, line costs, setups from one board to the others, etc.) is randomly created. The creation of the initial random population is performed as in the Heuristic 1 by randomly assigning numbers representing the assembly lines to the string. The evaluation of the population is performed as described previously, by randomly assigning boards belonging to groups of user defined size, then performing the multistart algorithm to find a local minimum sequence within a group and between groups. After the objective function value is determined for the entire population, fitness is calculated and the best element within a population is copied in another data structure and kept for later comparison.

The termination criteria is identical to the one applied in Heuristic 1, where a combination of the minimum number of generations, the maximum number of consecutive bests, and the maximum number of generations is used to end a GA run. Selection according to fitness is then performed as mentioned previously and a new population (or the intermediate stage in the generation cycle) is created.

The major contrast between this heuristic and Heuristic 1 is that the population is not always evaluated after the crossover operation. This allows more significant changes in the population members since both operations (crossover and mutation) are performed consecutively without re-evaluating the population member's fitness. Therefore, the probability of a bit in a population member with a good fit being changed is much higher. Consequently, the area investigated by the algorithm in the solution space is augmented. The drawback is that the mutation may unnecessarily break some high-quality schema created at the crossover. The gain with this heuristic is that it considerably speeds up the computation.

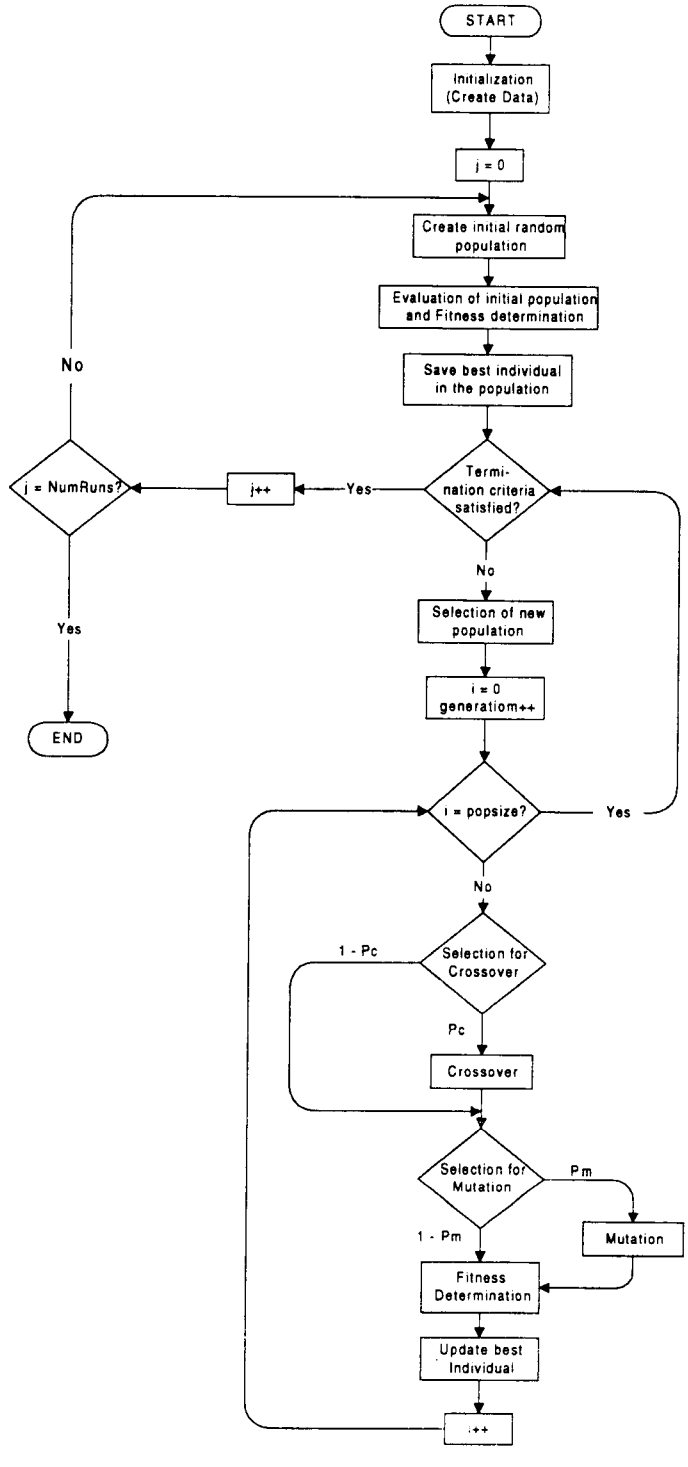


Figure 13 - Flowchart Heuristic 2

Therefore, as in Heuristic 1, a member of the new population is chosen to be created by crossover if a uniformly distributed random number in the range 0 to 1 is greater than or equal to a user defined probability of crossover (*Pc*). Two parents are chosen according to their fitness and the new member will be equal to the best member in this set (parents or children). Following, the roulette wheel routine is used and the probability of mutation (*Pm*) will determine if the same member will be subject to mutation. After the new population is created, fitness is determined and if the best member in the new population is better than the former best, it is copied in the data structure. The termination criteria is checked and if not reached a new generation is initiated following all the steps described previously. After the user defined maximum number of runs is reached the process is terminated and the best assignment along with all the concerning information is printed out.

Heuristic 3

Figure 14 shows the flowchart of Heuristic 3. This heuristic follows the Genetic Programming Paradigm (Koza 1994) pattern, known as Simple Genetic Algorithm (SGA), where a member of the new population can be created by either crossover, mutation, or reproduction. The steps concerning to the creation of the initial random population, the evaluation of each population member (including fitness determination), the termination criteria, and the selection of the members for the new generation are identical to Heuristics 1 and 2.

The main difference between this heuristic and the others is that only one genetic operation is chosen in the creation of a member of the new population. This is performed by selecting the genetic operation probabilistically, using the same roulette wheel routine. Therefore, crossover will be used in the creation of the member of the new population if a uniformly distributed random number in the range of 0 and 1 is greater than or equal to a user defined probability of crossover (*Pc*). Mutation will be performed if the random number is less than or equal to a user defined probability of mutation (*Pm*).

If non of the alternatives is performed (the random number lies in the 1-*Pc* - *Pm* interval) the member of the former population will be reproduced in the new population.

Crossover and mutation are performed as in the previous heuristics and the best between parent and child is reproduced in the new population. After the new population is created, fitness is determined and if the best member in the new population is better than the former best, it is copied in the data structure. The process is ended after the termination criteria is reached. The output is the same as in the other Heuristics.

Example Problem

Heuristic 1 is used to solve a small problem as an example to illustrate the functionality of the procedure and the way in which it simultaneously addresses the issues of assigning boards to lines and creating a sequence of boards in each line. The example shows the creation of one generation by performing crossover and mutation. The problem deals with a group of 8 boards which have to be assigned to 3 lines. The time horizon is 1 week.

The setup time matrix (number of changeovers times the required time for a changeover) is shown in Table 1. For simplification, the time spent on a feeder changeover was made equal to 1 time unit for all boards, independent of board types and lines. The assembly times (runtimes) of each board in each line as well as the weekly demand of each board are shown in Table 2. The line costs are shown in Table 3. For simplification, all lines have the same capacity, which is sufficient for assembling one third of the boards in a week. The population size (popsize) is 5, therefore there will be 5 members in each population. The probability of crossover (*Pc*) is 0.7 and the probability of mutation (*Pc*) is 0.3.

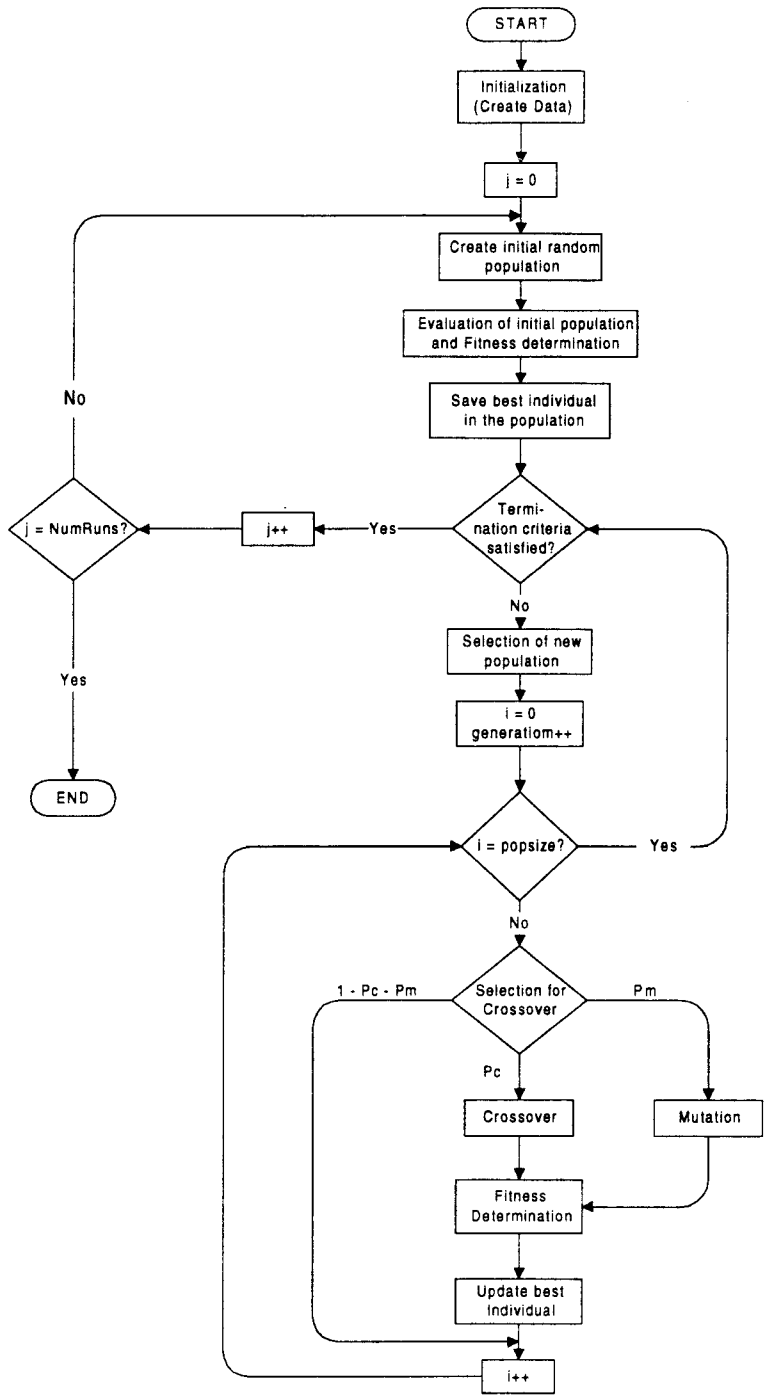


Figure 14 - Flowchart Heuristic 3

Boards	0	1	2	3	4	5	6	7
0		23	48	38	6	30	20	37
1	31		19	8	12	22	41	26
2	50	38		9	33	25	4	35
3	26	8	48		46	35	16	22
4	4	49	35	8		42	30	10
5	9	41	24	8	26		21	14
6	29	35	38	37	24	7		42
7	2	26	34	22	6	48	47	

Table 1 - Setup time matrix

Boards	Runtime			Demand
	Lines			
	0	1	2	
0	9	3	8	80
1	9	10	10	74
2	7	4	3	1
3	3	9	1	44
4	4	1	7	57
5	1	1	10	35
6	3	3	6	30
7	7	9	8	14

Table 2 - Manufacturing data

Line	Cost/min
0	3.0
1	1.0
2	5.0

Table 3 - Line Costs

The initial population and the number of boards assigned per line are shown in Table 4. After the assignment, boards allocated in a line are randomly assigned to groups. In this example the maximum group size was set to 5 and in the initial population

all groups formed per line were smaller than 5. Therefore, there was only one group per line for each assignment. This is due to the size of the problem dealt in this example and the randomness of the algorithm. There might be members in future generations that may have more than one group per line.

Pop. Member	Line Assignment	Num. Boards per line		
		0	1	2
0	1,1,1,2,0,1,0,2	2	4	2
1	0,2,1,0,2,2,1,1	2	3	3
2	2,0,2,1,0,1,0,2	3	3	2
3	1,1,0,2,0,0,2,0	4	2	2
4	2,0,1,0,2,2,2,1	2	2	4

Table 4 - Initial population

Table 5 shows the results obtained after evaluation of the first member in the population (member 0). The multistart sequence algorithm is used to find the minimum sequence in each group. For example, in line 0, boards 6 and 4 are assigned to a group. The sequencing routine finds a local minimum setup (in this case it finds the optimal sequence) by first starting with board 6 and following the setup steepest descent path. In this case, since there is only one more board in the group, it gets the number of changeovers if changing from board 6 to board 4. It can be seen in Table 1 that this number is 24. Then this procedure is repeated by starting with board 4. The minimum setup found in this path is 30. Therefore, the best sequence if boards 4 and 6 are in the same group, is starting with board 6 and then changing to board 4. This, along with the same information for the other lines is shown in Table 5.

The runtime costs (assembly costs) are calculated after the assignment of the boards for each line. Demand times the runtime to assemble boards in a specific line is used to calculate the total runtime in that line. Therefore, for member 0, having boards 6 and 4 assigned to line 0 would result in a total runtime of 318 time units. Total costs are

calculated by multiplying the total manufacturing time (assembly plus setup times) times the line costs. Tables 6,7,8 and 9 show these values for population members 1,2,3 and 4 respectively.

Member 0						
Line	Boards Assigned	Min Sequence	Tot. Setup	Assembly Times	Assembly Costs	Total Costs
0	6,4	6,4	24	318	954	1026
1	0,1,2,5	5,0,1,2	51	1019	1019	1070
2	3,7	3,7	22	156	780	890
Total			97	1493	2753	2986

Table 5 - Evaluation member 0

Member1						
Line	Boards Assigned	Min Sequence	Tot. Setup	Assembly Times	Assembly Costs	Total Costs
0	0,3	3,0	26	852	2556	2634
1	2,6,7	7,2,6	38	220	220	258
2	1,4,5	1,4,5	54	1489	7445	7715
Total			118	2561	10221	10607

Table 6 - Evaluation member 1

Member 2						
Line	Boards Assigned	Min Sequence	Tot. Setup	Assembly Times	Assembly Costs	Total Costs
0	1,4,6	1,4,6	42	984	2952	3078
1	3,5	5,3	8	431	431	439
2	0,2,7	2,7	37	755	3775	3960
Total			87	2170	7158	7477

Table 7 - Evaluation member 2

Member 3						
Line	Boards Assigned	Min Sequence	Tot. Setup	Assembly Times	Assembly Costs	Total Costs
0	2,4,5,7	2,5,7,4	45	368	1104	1239
1	0,1	0,1	23	980	980	1003
2	3,6	3,6	16	224	1120	1200
Total			84	1572	3204	3442

Table 8 -- Evaluation member 3

Member 4						
Line	Boards Assigned	Min Sequence	Tot. Setup	Assembly Times	Assembly Costs	Total Costs
0	1,3	1,3	8	798	2394	2418
1	2,7	7,2	34	130	130	164
2	0,4,5,6	6,5,0,4	22	1569	7845	7955
Total			64	2497	10369	10537

Table 9 - Evaluation member 4

The fitness values and the cumulative fitness of the initial population are determined and shown in Table 10. The cumulative fitness is used in the roulette wheel routine to select individuals for the next intermediate stage. The probability of selection (a uniformly distributed random number between 0 and 1) is given and members of the initial population are reproduced in the mating pool as described previously. Table 11 shows these values and the new population after selection. It can be seen that the best fit members are likely to be reproduced more often than members with poor fit.

Member	Tot. Cost	Fitness	Cum. Fitness
0	2986	0.423718	0.423718
1	10607	0.000000	0.423718
2	7477	0.174024	0.597743
3	3442	0.398365	0.996108
4	10537	0.003892	1.000000

Table 10 - Fitness values for initial population

Member	Line Assignment	Cum. Fitness	Prob. of Selection
0	2,0,2,1,0,1,0,2	0.597743	0.471584
1	1,1,0,2,0,0,2,0	0.996108	0.978457
2	1,1,1,2,0,1,0,2	0.423718	0.235642
3	1,1,0,2,0,0,2,0	0.996108	0.716532
4	1,1,1,2,0,1,0,2	0.423718	0.087957

Table 11 - New population after selection

After selection, parents are chosen according to fitness for the crossover operation. Table 12 shows an example where the crossover probability (a uniformly distributed random number) determined that members 1 and 0 would crossover in the formation of a member of the new population. The crossover point was also randomly generated and was set to after the first bit. Table 12 also shows the chromosome (string) of the parents as well as the one of the offspring generated after crossover.

Crossover Prob.	Parent Chosen	Parent Chromosome	Offspring Chromosome
0.769066	1	1,1,0,2,0,0,2,0	1,0,2,1,0,1,0,2
0.410718	0	2,0,2,1,0,1,0,2	2,1,0,2,0,0,2,0

Table 12 - Example of crossover selection

Table 13 and 14 show the evaluation of offspring 1 and 2 respectively. The process used here is identical to the one mentioned before. The results show that the best member among the parents and the offspring is member 3 of the previous population (or member 1 after selection), therefore it is reproduced in the new population.

Offspring 1						
Line	Boards Assigned	Min Sequence	Tot. Setup	Assembly Times	Assembly Costs	Total Costs
0	1,4,6	1,4,6	42	984	2952	3078
1	0,3,5	5,3,0	34	671	671	705
2	2,7	7,2	34	115	575	745
Total			110	1770	4198	4528

Table 13 - Evaluation of offspring 1

Offspring 2						
Line	Boards Assigned	Min Sequence	Tot. Setup	Assembly Times	Assembly Costs	Total Costs
0	2,4,5,7	2,5,7,4	45	368	1104	1239
1	1	1	0	740	740	740
2	0,3,6	3,6,0	45	864	4320	4545
Total			90	1972	6164	6524

Table 14 - Evaluation of offspring 2

The mutation operation is performed in the same fashion as the crossover operation. The new generation is finished after the creation of all members of the new population. Table 15 shows the results of the new generation created after selection, crossover, mutation and reproduction along with the new cumulative fitness. Notice the formation of the schema 1,1,0,2,0,*,*,* and how the solutions with this schema have the highest probability of being selected for the next generations.

Member	Line Assignment	Cum. Fitness
0	1,1,0,2,0,1,0,2	0.391021
1	1,1,0,2,0,0,2,0	0.820858
2	1,1,0,2,0,0,2,0	0.820858
3	1,1,0,2,0,1,0,2	0.825894
4	2,1,2,2,1,0,2,0	1.000000

Table 15 - New population after the first generation

Only one run with 5 generations of the algorithm was performed and the best assignment was found to be 1,1,0,2,0,1,1,0 with a total cost of 2486. This result is 76% better than the worst assignment found in the initial population. Notice that the schema formed in the first generation lasted for all other generations.

EXPERIMENTAL DESIGN

The three heuristics presented here make use of a set of parameters which are significant for the GA search, they are:

- Number of runs (Runs)
- Population Size (Popsize)
- Number of generations with the same best solution (BestRepet)
- Probability of Crossover (PC)
- Probability of Mutation (PM)
- Number of bits mutated (MutElem)

The number of runs establishes how many initial random populations are investigated during the search. Therefore, when increased, the number of subsets of the solution space investigated is increased, and the higher the probability of finding a better solution. The drawback is that the computation time is also increased. The population size has similar effects in the search. When increased, more points in the solution space are investigated in every generation which may lead to a better search. A shortcoming is also the increase in computation time.

The number of generations with the same best determines the condition to stop the search. Therefore, a small number can make the search being trapped in a local optimum. A large number reduces the efficiency of the search and increases the computation time. The amount of variation in the genetic material inserted by mutation is determined by the number of bits to be mutated in a member. If large, it can result in the breaking of good quality schema, thus in a poor search. It also increases the computation time when increased.

As mentioned previously, the crossover and mutation operations are the main factors in driving the search to a better solution. They allow the GA to investigate several

different points in the solution space by making extreme (crossover) and moderate (mutation) changes in the population members. Therefore, the probability of crossover and the probability of mutation are extremely important in determining the efficiency and effectiveness of the search.

The previous research in GA does not explore and define the set of values for these parameters in two layered problems, some suggestions maybe found (Srinivas and Patnaik 1994) but they were not investigated. It is also assumed that these parameters have different effects on each heuristic. Therefore, an experimental design was performed in order to determine the best set of values for each heuristic. This serves in customizing the parameters for every heuristic in order to achieve the best performance.

Two experiment designs were performed. Initially, a 2^{K-P} (K = 6 and P = 1) fractional factorial design was used to determine which parameters were likely to be important. Analysis of variance was used to confirm this interpretation. This design was chosen since the experiment deals with many factors and it provides a reasonable number of runs. Because there are only two levels for each factor, we must assume that the response is approximately linear over the range of the factor levels chosen.

This "screening test" was performed for each heuristic using the same levels for each factor and the same experiment runs. Two response values were taken, the objective function value of the best solution found (TotCost) and the computation time (CompTime) taken to find that solution. Table 16 shows a summary of the design and the parameter values used as well as the coding of the factors. The values for each level were chosen according to intuitive knowledge of the heuristic performance and with the concern of choosing values that could be used in the following experiment. Tables A1, A2, A3 (appendix A) show the runs and the results obtained for each run as well as the configuration of the best solution found for heuristic 1,2, and 3 respectively. Figures A1 and A2 show the plot of the two response values for each heuristic, TotCost and CompTime respectively.

Item	Value
Number of experimental factors	6
Number of responses	2
Number of runs	32
Number of blocks	1
Number of centerpoints per block	0
Error degrees of freedom	10

Factors	Low	High
Runs	40	60
Popsize	50	100
BestRepet	6	10
MutElem	6	10
PC	.6	.8
PM	.1	.2

Source	Identification
GA Runs	A
Popsize	B
BestRepet	C
MutElem	D
PC	E
PM	F

Table 16 - Experiment design summary

After the screening test was performed and the significant parameters were identified for each heuristic, a new experiment (a regression design) was done in order to establish the relationship between the response and the independent variables. In other words, the screening test and the analysis of variance assisted on identifying which factors were important, and the regression analysis was used to build a quantitative model relating the important factors to the response. A central composite design (rotatable and orthogonal) was applied for the regression analysis. The number of factors used in this step differs for each heuristic since it depends on the significance of each factor for that heuristic determined in the first experiment.

The following sections present the analysis performed in both experiments for each heuristic.

Heuristic 1

The runs performed and the corresponding set of parameters are shown in Table A1. It also shows the values of the two responses as well as the configuration of the best solution. The initial analysis of variance regarding to TotCost with all factors and the two-factor interactions is shown in Table A4. The ANOVA Table partitions the variability in TotCost into separate pieces for each of the effects. It then tests the statistical significance of each effect by comparing the mean square against an estimate of the experimental error. In this case, 2 effects (B and D) have P-values less than 0.05, indicating that they are significantly different from zero at the 95% confidence interval. The R-square statistic indicates that the model as fitted explains 74% of the variability in TotCost.

The same analysis was applied for the two significant main effects in order to eliminate the variability arising from the nuisance sources. Table A5 shows the new ANOVA Table where the P-values indicate that the 2 main effects (B and D) are significant indeed. This indicates that these two factors should be used in the regression analysis. The residual analysis, performed by the normal probability plot of the residuals and the plot of the residuals against the predicted values, did not reveal anything particularly troublesome. Therefore it can be assumed that the model is adequate and that the error terms are normally and independently distribute with constant variance. Both plots are shown in Figures A3 and A4. Figure A5 shows the main effects plot for TotCost of B and D.

Table A6 shows the initial analysis of variance regarding to CompTime with all factors and the two-factor interactions. In this case, 6 effects (main effects A, B, D, and F and two-factor interaction AB, and AF) have P-values less than 0.05, indicating that they are significantly different from zero at the 95% confidence interval. The R-square statistic indicates that the model as fitted explains 99.8% of the variability in CompTime. The same analysis was applied for these factors and is shown in Table A7.

The P-values indicate that these effects remain significant after eliminating the nuisance variables. This indicates that they should be used in the regression analysis.

The residual analysis did not reveal any problem. The normal probability plot of residuals and the plot of residuals against the predicted values are shown in Figure A6 and A7 respectively. To assist in the practical interpretation of this experiment, Figures A8 and A9 present plots of the four main effects, Figures A10 and A11 the AB and AF interactions respectively.

Item	Value
Number of experimental factors	4
Number of responses	2
Number of runs	36
Number of blocks	1
Number of centerpoints per block	12
Error degrees of freedom	21

Source	Identification
GA Runs	A
Popsize	B
MutElem	C
PM	D

Table 17 - Summary Response surface design heuristic 1

The analysis of the screening test reveals that factors A, B, D, and F and their interactions should be investigated in the regression analysis. Therefore, a 2^4 + *star* central composite design was performed to examine the magnitude and direction of these factors. Table 17 shows a summary of this design and the new coding used. The runs performed and the corresponding set of parameters are shown in Table B1 (appendix B). It also shows the values of the two responses as well as the configuration of the best solution.

Table B2 shows the initial analysis of variance regarding to TotCost with all factors and the two-factor interactions. In this case only 1 effect (main effect B) has P-

values less than 0.05. The R-square statistic indicates that the model as fitted explains 38.2% of the variability in TotCost. The lack-of-fit test is not significant, therefore the selected model is adequate to describe the observed data. The same analysis was applied for this factor along with some other factors with P-values less than 0.3 (two-factor interactions AA, AC, BB, CD) in order to assure that the results were not caused by variability inserted by nuisance variables. The criteria of checking some marginal factors was used in the response design in order to obtain a secure model. Table B3 shows this analysis and it can be seen that factor B remained significant while the other factors remained non-significant. A third analysis was performed only for factor B and is shown in Table B4. The R-square statistic indicates that the model as fitted explains 14.5% of the variability in TotCost and the lack-of-fit test remained not significant. Therefore the model fitted adequately explains the data but weakly explains the variability in TotCost.

The residual analysis did not reveal any problem. The normal probability plot of residuals and the plot of residuals against the predicted values are shown in Figure B1 and B2 respectively. Figure B3 presents a plot of the B main effect. Notice that the population size has positive main effect; that is, increasing the variable moves the average of TotCost downward.

Following the regression equation which has been fitted to the data. The equation corresponds to the plot of the Popsize main effect.

$$\text{TotCost} = 8717.4 - 16.595 * \text{Popsize}$$

Table B5 shows the initial analysis of variance regarding to CompTime with all factors and the two-factor interactions. In this case 8 effects (three main effects A, B, and D and five two-factor interactions AB, AD, BD, CD, and DD or D-squared) have P-values less than 0.05 . The R-square statistic indicates that the model as fitted explains 99.9% of the variability in TotCost. The lack-of-fit test is significant, therefore some deterministic pieces of data cannot be described by the model. Main effect C and the interaction AA were included in the next analysis since they have marginal significance.

The analysis of the revised model is shown in Table B6. Notice that the interaction AA remained non-significant. A new analysis is then performed without the AA interaction and is shown in Table B7. Notice that all factors remained significant except by C which has very marginal P-value. For this reason it was kept in the model. The R-square statistic indicates that the model as fitted explains 99.9% of the variability in CompTime and the lack-of-fit test remained significant.

The residual analysis did not reveal any problem. The normal probability plot of residuals and the plot of residuals against the predicted values are shown in Figures B4 and B5 respectively. Figures B6, and B7 present plots of A, B, C, D main effects. Figures B8, B9, B10, and B11 show the effects of the AB, AD, BD, and CD interactions. Notice that all the main effects have negative effect; that is, their increase moves the average of CompTime upward.

Following the regression equation which has been fitted to the data. The estimated response surface plots of the AB, AD, BD, and CD interactions are shown in Figures B12, B13, B14, and B15 respectively.

$$\text{CompTime} = 21.3559 - 0.533 * \text{Runs} - 0.0454167 * \text{Popsize} - 1.09115 * \text{MutElem} - 127.321 * \text{PM} + 0.0421675 * \text{Runs} * \text{Popsize} + 2.82875 * \text{Runs} * \text{PM} + 0.8135 * \text{Popsize} * \text{PM} + 8.70625 * \text{MutElem} * \text{PM} - 173.792 * \text{PM}^2$$

Heuristic 2

The runs performed on the screening test and the corresponding set of parameters is shown in Table A2. The initial analysis of variance regarding to is shown in Table A8. In this case, 1 effect (the two-factor interaction AD) have P-values less than 0.05, indicating that they are significantly different from zero at the 95% confidence interval. The main effects D and F and the CD interactions have marginal P-values and therefore

are included in the following analysis. The R-square statistic indicates that the model as fitted explains 74% of the variability in TotCost.

The same analysis was applied for D, F, AD and CD effects and is shown in Table A9. The CD interaction remained non-significant and the other values (with the exception of F which is marginal) are significant. Therefore, factors A, D and F and their interactions should be investigated by the regression analysis. The residual analysis, performed by the normal probability plot of the residuals and the plot of the residuals against the predicted values, did not reveal any problem. Both plots are shown in Figures A12 and A13. Figure A14 shows the main effects plot for TotCost of D and F and Figure A15 shows the plot of the AD interaction.

Table A10 shows the initial analysis of variance regarding to CompTime. In this case, 6 effects (main effects A, B, and E and two-factor interaction AB, BF, and EF) are significant. The R-square statistic indicates that the model as fitted explains 98.6% of the variability in CompTime. The same analysis was applied for these factors and is shown in Table A11. The P-values indicate that these effects remain significant after eliminating the nuisance variables. Therefore factors A, B, E and F and their interactions should be investigated by the regression analysis

The residual analysis did not reveal any problem. The normal probability plot of residuals and the plot of residuals against the predicted values are shown in Figure A16 and A17, respectively. Figures A18 and A19 present plots of the three main effects Figures A20, A21 and A22 show the plots of the AB, BF and EF interactions.

The analysis of the screening test reveals that factors A, B, D, E and F should be used in the regression analysis. Therefore, a 2^{5-1} + *star* central composite design was performed to examine the magnitude and direction of these factors. Table 18 shows a summary of this design and the new coding used. The runs performed and the corresponding set of parameters are shown in Table B8. It also shows the values of the two responses as well as the configuration of the best solution.

Table B9 shows the initial analysis of variance regarding to TotCost with all factors and the two-factor interactions. In this case only 2 effects (main effect B and two-factor interaction BD) are significant. The R-square statistic indicates that the model as fitted explains 73.5% of the variability in TotCost. The lack-of-fit test is not significant, therefore the selected model is adequate to describe the observed data.

Item	Value
Number of experimental factors	5
Number of responses	2
Number of runs	36
Number of blocks	1
Number of centerpoints per block	10
Error degrees of freedom	15

Source	Identification
GA Runs	A
Popsize	B
MutElem	C
PC	D
PM	E

Table 18 - Summary Response surface design heuristic 2

The same analysis was applied for this factor along with two-factor interactions AD, BB, CC, CD, CE, and EE which have marginal values. Table B10 shows this analysis and it can be seen that effects of B and BD remained significant while the other factors were not significant. A third analysis was performed only for factor B and the BD interaction and is shown in Table B11. The R-square statistic indicates that the model as fitted explains 29% of the variability in TotCost and the lack-of-fit test remained not significant. Therefore the model fitted adequately explains the data but does not explains well the variability in TotCost.

The residual analysis did not reveal any problem. The normal probability plot of residuals and the plot of residuals against the predicted values are shown in Figure B16 and B17 respectively. Figure B18 presents a plot of the BD interaction. Notice that the

population size has also positive main effect in heuristic 2. The estimated response surface of the BD interaction is shown is Figure B19. Following the regression equation which has been fitted to the data:

$$TotCost = -2296.46 + 134.592 * Popsize + 16143.7 * PC - 215.25 * Popsize * PC$$

Table B12 shows the initial analysis of variance of the response design regarding to CompTime. In this case 5 effects (three main effects A, B, and E and two two-factor interactions AB, and CC) are significant. The R-square statistic indicates that the model as fitted explains 99.7% of the variability in CompTime. The lack-of-fit test is not significant, therefore the model adequately describes the data. Factors AE, BE and EE were included in the following analysis since they have marginal P-values.

The analysis of the revised model is shown in Table B13. Notice that the interactions AE and EE remained non-significant. A new analysis is then performed without the these factors and is shown in Table B14. The factor B turned out to be non-significant. The anova Table in B15 shows the final result with the effects of A, B, E, AB, and CC being significant. The R-square statistic indicates that the model as fitted explains 99.5% of the variability in CompTime and the lack-of-fit test remained not significant.

The residual analysis did not reveal any irregularity. The normal probability plot of residuals and the plot of residuals against the predicted values are shown in Figure B20 and B21 respectively. Figures B22 and B23 present plots of the A, B, E main effects. Figure B24 shows the effects of the AB. Notice that, like in heuristic 1, all main effects have negative effect. Figure B25 shows the estimated response surface of the AB interaction. Following the regression equation which has been fitted to the data:

$$CompTime = 12.5362 - 0.299271 * Runs - 0.064925 * Popsize - 4.83875 * MutElem + 96.2583 * PM + 0.0321475 * Runs * Popsize + 0.302422 * MutElem^2$$

Heuristic 3

The runs performed on the screening test and the corresponding set of parameters and results is shown in Table A2. The initial analysis of variance regarding to TotCost is shown in Table A12. In this case, 2 effects (the main effect B and the two-factor interaction CF) are significant. The two-factor interactions AB, AD, BC and BD are included in the following analysis since they have marginal P-values. The R-square statistic indicates that the model as fitted explains 74% of the variability in TotCost.

The analysis of the revised model is shown in Table A13. The AD and BD interactions remained non-significant and the other values (with the exception of F which is marginal) are significant. The model is revised again, Table A14 shows the new anova. Notice that with more degrees of freedom for the error term the AB interaction is not significant. Therefore factors B, C and F and their interactions should be investigated by the regression analysis. The residual analysis, performed by the normal probability plot of the residuals and the plot of the residuals against the predicted values, did not reveal any problem. Both plots are shown in Figures A23 and A24. Figures A25, A26 and A27 show plots of the B main effects for TotCost and of the BC and CF interactions respectively.

Table A15 shows the analysis of variance of the screening test regarding to CompTime. In this case, 7 effects (main effects A, B, E, and F and two-factor interaction AB, AF, and BF) are significant. The R-square statistic indicates that the model as fitted explains 99.8% of the variability in CompTime. The same analysis was applied for these factors and is shown in Table A16. The P-values indicate that these effects remained highly significant after eliminating the nuisance variables.

The residual analysis did not reveal any problem. The normal probability plot of residuals and the plot of residuals against the predicted values are shown in Figure A28 and A29 respectively. Figures A30 and A31 present plots of the four main effects Figures A32, A33 and A34 the effects of the AB, AF and BF interactions.

The analysis of the screening test reveals that factors A, B, D, E and F and their interactions should be used in the regression analysis. Therefore, a 2^{5-1} + *star* central composite design was performed to examine the magnitude and direction of these factors. Table 19 shows a summary of this design and the new coding used. The runs performed and the corresponding set of parameters are shown in Table B16. It also shows the values of the two responses as well as the configuration of the best solution.

Table B17 shows the analysis of variance of the response design regarding to TotCost. In this case only 2 effects (main effect B and two-factor interaction CC) are significant. The R-square statistic indicates that the model as fitted explains 80.2% of the variability in TotCost. The lack-of-fit test is not significant, therefore the selected model is adequate to describe the observed data.

The same analysis was applied for this factor along with main effect D and two-factor interactions AB, BB, and BE which have marginal values. Table B18 shows this analysis and it can be seen that effects of B, D and CC are. A third analysis was performed only for these factors and is shown in Table B19. All of them remained significant. The R-square statistic indicates that the model as fitted explains 61.6% of the variability in TotCost and the lack-of-fit test remained not significant. Therefore the model fitted adequately explains the data and adequately explains the variability in TotCost.

Item	Value
Number of experimental factors	5
Number of responses	2
Number of runs	36
Number of blocks	1
Number of centerpoints per block	10
Error degrees of freedom	15

Source	Identification
GA Runs	A
Popsize	B
BestRepet	C
PC	D
PM	E

Table 19 - Summary Response surface design heuristic 3

The residual analysis did not reveal any problem. The normal probability plot of residuals and the plot of residuals against the predicted values are shown in Figures B26 and B27 respectively. Figure B28 presents a plot of the B and D main effects. Notice that the population size has negative effect in heuristic 3 while the probability of crossover has positive effect. The estimated response surface of interactions BD, BC and DC are shown is Figures B29, B30, and B31 respectively. Following the regression equation which has been fitted to the data:

$$\text{TotCost} = 14802.9 - 40.485 * \text{Popsize} - 1390.71 * \text{BestRepet} + 3405.42 * \text{PC} + 86.9193 * \text{BestRepet}^2$$

Table B20 shows the initial analysis of variance of the response design regarding to CompTime. In this case 13 effects (four main effects A, B, D and E and nine two-factor interactions AB, AC, AE, BB, BD, BE, CD, DE, EE) are significant. The R-square statistic indicates that the model as fitted explains 99.9% of the variability in CompTime. The lack-of-fit test is not significant, therefore the model adequately describes the data. Factors AA, and DD were included in the following analysis since they have marginal P-values.

The analysis of the revised model is shown in Table B21. Notice that the AA interaction remained non-significant while the DD interaction became significant with the increase of degrees of freedom on the error term. The AA interaction was taken off and a new analysis was performed and is shown in Table B22. All the effects remained significant. The R-square statistic indicates that the model as fitted explains 99.9% of the variability in CompTime and the lack-of-fit test remained not significant.

The residual analysis did not reveal any problem. The normal probability plot of residuals and the plot of residuals against the predicted values are shown in Figures B32 and B33 respectively. Figures B34 and B35 present plots of the A, B, D and E main effects. Figures B36, B37 , B38, B39, B40, B41, and B42 present plots of the AB, AC, AE, BD, BE, CD, and DE interactions respectively. Figures B43, B44, B45, B46, B48, and B49 show the estimated response surface of the AB, AC, AE, BD, BE, CD, and DE interactions. Following the regression equation which has been fitted to the data:

$$\text{CompTime} = 58.9618 - 1.28525 * \text{Runs} - 0.87945 * \text{Popsize} - 2.67188 * \text{BestRepet} + 121.54 * \text{PC} - 636.242 * \text{PM} + 0.0377275 * \text{Runs} * \text{Popsize} - 0.0662187 * \text{Runs} * \text{BestRepet} + 7.15375 * \text{Runs} * \text{Pm} + 0.0030625 * \text{Popsize}^2 - 1.01075 * \text{Popsize} * \text{PC} + 6.3285 * \text{Popsize} * \text{PM} + 8.54688 * \text{BestRepet} * \text{PC} - 64.0938 * \text{PC}^2 - 393.875 * \text{PC} * \text{PM} + 1298.12 * \text{PM}^2$$

APLICATION OF THE HEURISTIC

Two examples were used to illustrate the functionality of the procedure in addressing the assignment of boards to lines. First the search engine was applied in a small problem with known optimum solution. Second the algorithm was used to solve a real problem encountered in the industry. In both cases the parameters were adjusted regarding to total costs since this is the main factor industry is concerned with. With the same purpose Heuristic 1 was used in both cases. This because it is believed that on large problems, where setup times are the main drive to better solutions, it will be the one that will provide the best results since it constantly attempts to improve setup times.

Small problem with known optimal

The purpose of this experiment was to test the quality of the solution found by the GA, by comparing it to a known optimal solution. Therefore, an optimal search algorithm (OSA) was implemented in order to allow the comparison with the results obtained with the GA implementation. This algorithm was coded using ANSI C and run on a Sun Sparc 20 Unix workstation.

The algorithm determined all possible permutations of boards to lines and all possible combinations in order to determine the best sequence per line. It proved to be very inefficient since it applies an explicit enumeration technique on the sequencing part, and therefore checks all points in the solution space, making its use in large sets of boards impractical. If the experiment would be concerned with efficiency, a branch-and-bound technique would be more appropriate since it would result in smaller computation times, although still unusable for large problems. However, since the GA investigates much less points than these other methods, it is expected to always have a better performance. Therefore, even if a branch-and-bound technique would result in better computation times, an efficiency comparison on a small problem would still be meaningless. Therefore, the following comparisons between the GA and the OSA

regarding computation time are taken only as an assessment of the GA's performance. These differences could be smaller if a branch-and-bound algorithm would be used instead of the OSA.

A set of 16 boards to be assigned to 3 lines was chosen as an appropriate problem for the test. The number of feeder changeovers between boards, demands per board, runtimes, capacity and costs for each line were randomly generated. The total number of points in this solution space cannot be accurately determined since the number of combinations for the sequencing depends on the number of boards assigned to a line. The total number of solutions can be estimated by the number of permutations multiplied by the factorial of the average number of boards assigned to a line. For this problem $3^{16} * 6!$ points (considering 6 as the average number of boards per line), which would result approximately in $3.1 * 10^{10}$ points to be investigated.

Heuristic 1 was used in 5 different runs keeping the same set of parameters and varying the random seed. The parameters were set according to the best result found with that heuristic (Table B1, run 30) except the number of runs which was set to 1. This provides the most severe condition for the GA where it has to find a near optimal solution in only one run. This method was chosen since it reduces the computation time and consequently demonstrates the efficiency of the algorithm. In order to better test the quality of the solution found by the GA, after the termination of each run the OSA was run a second time in order to count the number of solutions found within the range between the GA and the optimal solution. This was made in order to determine how many points were within that range (i.e. the GA would not prove to be efficient if 70% of all points were within the 3% range from the optimal and the algorithm found one of them). The results of each run and the optimal solution found with the OSA are shown in Table 20.

Run	OSA		GA			
	CompTime (hours)	TotCost	CompTime (sec.)	TotCost	%diff. TotCost	Solutions within this range
01	7.58	45,096	4.18	46,493	3.1	2
02	7.58	45,096	2.09	46,641	3.4	2
03	7.58	45,096	2.33	45,096	0.0	1
04	7.58	45,096	6.23	47,319	4.9	16
05	7.58	45,096	1.80	47,486	5.3	47

Table 20 - Summary of results for the small problem with known optimal

The results show the efficiency of the Genetic Algorithm in finding a near optimal solution. On the first and second run the solution found in a fraction of computation time were 3.1% off the optimal solution. The re-run proved that there were only 2 points within the 3.1% range, the optimal solution and the one found by the GA. The third run shows that the algorithm found the optimal solution. The percentage of variation on all runs was 5.3%. In the worst case the GA solution was among 47 other points which is still an infinitesimal fraction of the total number of points in the solution space.

With the knowledge of the performance of the GA in finding a near optimal solution a second experiment was performed. In this case the problem size (16 boards) was kept and the problem parameters (runtimes, demands, capacities, costs, etc.) were randomly changed. This experiment was basically concerned in determining the difference between the optimal solution and the one found by the GA. Re-runs of the OSA were not performed at this time. Table 21 shows a summary of the results.

Run	OSA		GA		
	CompTime (hours)	TotCost	CompTime (sec.)	TotCost	%diff. TotCost
01	7.61	488,150	12.81	488,150	0.00
02	7.57	57,533	7.68	59,119	2.76
03	7.59	81,787	1.09	85,276	4.27
04	7.57	37,665	3.46	37,752	0.23
05	7.57	51,765	6.29	51,798	0.06

Table 21 - Summary of results for the small problem (2nd experiment)

Again the results found by the GA were very close to the optimal solution. It can be noticed by comparing the experimental runs from the previous chapter and the ones performed in this section, that the number of GA runs strongly influences the computation time. Adding to that, the computation time was also improved due to the use of a faster machine.

Industry Problem

As the problem of assigning boards to lines attempting to setup reduction through sequencing is proven NP-hard, a large problem would require non-polynomial (exponential) time to solve for their optimal solution. Therefore, the test described in the previous section cannot be applied on a large problem. Moreover, although the results show the ability of the GA implementation in finding near-optimal solutions, there is no guarantee that an optimal solution for a large problem can be found within a reasonable (polynomial) computation time. Hence, the underlying assumption in solving large problems is that since the GA based algorithm applied to the small problem was able to identify the best solution within a short search span and under the most severe conditions, it is capable of identifying the best solution that is presumably close to the optimal solution, although this cannot be proven.

Therefore, with the heuristic performing efficiently and giving promising results to the sample problems, it was applied to a large industry sized problem. This would be the ultimate test for the heuristic since it is now used to determine whether or not real savings in cost and increasing of machine utilization can be realized by applying it to the manufacturing floor of an industry engaged in batch production of parts. With that purpose Tektronix Inc., Beaverton, Oregon, was the industrial partner in this research and hence data pertaining to its PCB manufacturing operations were used to test the heuristic algorithm.

The manufacturing system can be characterized as a very high mix, low volume environment, where batch sizes vary from 1 to 100 boards. It uses a total of 3 surface mount lines to produce 905 boards expressing more than 1400 board part numbers. More than 7000 different components are used on the assembly of these boards.

The average cost per unit of line runtime, representative of the total cost incurred in operating one line, was determined by calculating the machine depreciation costs and the labor involved in the manufacturing. Costs related to building facilities, administrative departments, and other, were ignored since were assumed to be the same for each line. The depreciation costs were calculated from information given for the purchase cost of the machine, the useful life of the machine and the internal rate of return. A Perl 5.0 script was used to automate this calculation allowing its use later on by the company.

The components changeover between boards was determined by comparing each board to all the other ones. The number of different parts encountered on this comparison was taken as the number of feeder changeovers. This was calculated as follows:

$$C_{ij} = \sum_{k=0}^{N} P_{ijk}$$

where:

i and j = boards

k = single component

N = total number of components in boards i and j

C_{ij} = total number of feeder changeovers between boards i and j

P_{ijk} = 0 if part k is found in board i and j, 1 otherwise

The information about the components were retrieved from Allegro CAD files and saved in binary files in order to speed up the search. This procedure was coded in ANSI C and run in a Sun Sparc 20 Unix workstation. A comparison algorithm using hash tables was implemented in order to reduce the computation time because of the large number of boards to be analyzed and compared. This algorithm consists of 3 Perl 5.0 scripts and an ANSI C program. The concern was to create tools that could be used in the future.

Information regarding demands and runtimes were retrieved from the company's database. An SQL (Structured Query Language) query written in Progress 8.2 was used to retrieve this information and, in order to allow the interface between the different servers a CGI (Common Gateway Interface) program written in Perl 5.0 was also implemented. This allows the use of the GA through a web browser using an HTML (Hyper Text Markup Language). Tables 22 and 23 show the line costs and placement times per line. It can be seen that line 3 has the fastest placement times and line 1 the slowest. The demand volume was assumed to be fixed and was taken for a time horizon of 12 months.

Moreover, another constraint was added so as to make the approach more realistic. In PCB manufacturing PCBs using the same raw board are often scheduled sequentially in order to reduce stencil setups and preparation on the printers. Therefore,

the demands of PCBs using the same raw board were added and only one board representing the whole group was used on the GA search.

Lines	Equipment	Labor	Total / year	Total /min	diff %
2	$ 285,337.48	$ 533,588.25	$ 818,925.73	$ 1.64	
1	$ 251,974.10	$ 567,661.57	$ 819,635.67	$ 1.64	0.09%
3	$ 305,566.44	$ 579,246.50	$ 884,812.95	$ 1.77	8%
total	$ 842,878.02	$ 1,680,496.33	$2,523,374.35	$ 5.05	

Table 22 - Line Costs - Industry problem

Lines	Placement Speed (parts/min)	diff. %
1	333	
2	428	+28.5
3	600	+80.2

Table 23 - Placement speed per line

After gathering all data, the algorithm was run in a Sun Sparc 20 Unix workstation. Heuristic 1 was again applied and in order to avoid a large computation time only one GA run was allowed. After the results were obtained, a simulation program was then implemented in ANSI C in order to measure the improvement gained with the new assignment. The input for the simulation was the current set of boards assigned to lines. Costs, demands, runtimes, capacity constraints, changeovers, etc. were the same used on the GA analysis. The simulation consists of finding a near optimal sequence for the boards fixed to lines, trying to simultaneously improve throughput and machine utilization. The same algorithmical steps from the second layer of the GA were used here. Table 24 shows an analysis of the results obtained with the GA.

Lines	% of demand			Total number of boards		
	Current	GA	%diff.	Current	GA	%diff.
1	27	16	-40.7	236	349	47.8
2	42	29	-30.9	364	316	-13.2
3	31	55	77.4	305	240	-21.3

Table 24 - Analysis of GA results - Industry Problem

The total time taken for the simulation was 3.4 hours and for the GA 6.4 minutes. The results show that the GA improved the total operating costs of the current assignment by 12% in only one GA run. It also provided a list with groups of boards that if scheduled as indicated will reduce feeder changeovers. It can be seen on Table 24 that in the current assignment boards and demands are fairly evenly distributed through the lines with line 2 being the one with the larger amounts. With the GA, the number of boards on line 3 was reduced by more than 21% and the summation of all demands assigned to that line increased by 77.4%. This means that boards with higher demands were assigned to that line which consequently reduces the number of times setups are required for changing from one board to another. At the same time, the number of boards assigned to line 1 increased by more than 47% and the total demand was reduced by 40.7%. This means that the boards with very low demand were assigned to that line incurring in constant setups. These results make sense since the GA attempted to reduce idle time of the line with higher production rates. Therefore, in order to improve the overall system, the GA sacrifices the line with lowest production rates by dedicating it to boards with low volumes.

CONCLUSION AND SUGGESTIONS FOR FURTHER RESEARCH

A model and a solution algorithm have been developed for effectively determining the near-best set of printed circuit boards to be assigned to surface mount lines. The problem was formulated as a Linear Integer Programming model attempting to setup reduction and increase of machine utilization while considering manufacturing constraints. Three heuristics using Genetic Algorithms were then developed in order to search for near optimal solution for the model. Previous research has not shown evidence of dealing with such a problem.

In order to determine the parameters with larger influence on the performance of each heuristic and to determine the best set of values for each of these parameters, two experimental designs were performed: a fractional factorial design and a central composite design.

The heuristic which is assumed to have the best performance was then tested on two different problem structures. The first compared the results obtained with the heuristic with the optimal known solution. The problem considered a set of 16 boards to be assigned to 3 lines. The optimal solution was obtained using an Optimal Search Algorithm (OSA) and after every Genetic Algorithm heuristic (GA) run the number of solutions between the optimal and the GA solution was obtained. The GA search algorithm proved to find a near optimal solution within a very small search span even when there were just a few other solutions in that range. The average distance between the optimal solution and the one obtained with the search algorithm was 3.3%. In some cases the algorithm found the optimal solution. The heuristic was also tested on 5 problems of the same size (16 boards assigned to 3 lines) varying the input data (demand, runtimes, number of feeder changeovers, etc.) where the optimal solution was also known. The average distance between the optimal solution and the one obtained with the GA was 1.5%. Again, In some cases the algorithm found the optimal solution.

The heuristic was also tested on another problem structure (an industry size problem), one having 905 boards representing 1400 board part numbers to be assigned to 3 lines. The results obtained from this further test were compared to a simulation of the current assignment. The assignment with the GA heuristic proved to reduce the total operating costs by 12%. This confirms the fact that the heuristic is efficient in determining a better solution for the system. A new constraint regarding PCBs using the same raw board was added which allowed the GA search to also reduce setups on the surface mount line printers. Moreover, the GA algorithm provided a list of small board groups resulting in minimum feeder changeovers. The problem structures cover a wide range of possible problems, thus proving the functionality of the algorithm in practice.

The above research can be further extended to investigate the effect of certain other parameters on the development of the model and the solution technique. One potential area could be the effect of component package size when determining the number of changeovers from one board to another. Depending on the package size one or more feeders have to be changed. This would increase the number of feeders to be changed and can possibly influence the final result.

Another area of extension is the determination of a fixed set of feeders on the machine feeder carrier. Once the assignment of boards is determined, the configuration of the feeder carrier should be divided into two sets: a configurable and a fixed set. The fixed set would be determine based on board usage, demands, size and slot constraints and should focus on minimizing feeder changeovers.

Another area of extension would be to improve the scheduling capabilities of the above research. Once the set of boards per line is determined, the problem of sequencing boards in a group and the groups should be dealt so as to minimize or maximize some measure of effectiveness. This problem is generally known as group scheduling.

Last, the above research could be extended by determining the distribution of parts among the machines within the surface mount line. These lines have usually a high

speed placement machine and other placement machines used for placing large and fine pitch components. The difference on the placement speed of these machines usually results in a work in process between them (depending on the number of components placed by each machine). Therefore, the distribution of components among these machines focusing on line balancing and on reducing time span or increasing throughput should be determined.

BIBLIOGRAPHY

Askin, R. G., Subramaniam, S. P., 1985, An economic approach to group technology configuration. *Technical Report, Industrial and Management Engineering - The University of Iowa. International Journal of Production Research,* Vol. 28, 1555-1572.

Askin, R. G., Chiu, K. S., 1990, A graph partitioning procedure for machine assignment and cell formation in technology.

Askin, R. G., Dror, M., and Vakharia, A. J., 1994, Printed circuit board family grouping and component allocation for a multimachine, open-shop assembly cell. *Naval Research Logistics*, Vol. 41, 587-608.

Barnea, A., and Sipper, D., 1993, Setup reduction in PCB automated assembly. *Computer Integrated Manufacturing Systems*, Vol. 6.

Ben-Arieh, D., and Dror, M., 1990, Part assignment to electronic insertion machines: two machines case. *International Journal of Production Research*, Vol. 28, No. 7, 1317-1327.

Bhaskar, G., and Narendran, T. T., 1996, Grouping PCBs for setup reduction: a maximum spanning tree approach. *International Journal of Production Research*, Vol. 34, No. 3, 621-632.

Burbidge, J. L., 1971, Production planning. *Heinemann*, London.

Burbidge, J. L., 1975, The introduction of group technology. *John Wiley & Sons*, New York.

Burbidge, J. L., 1989, Production flow analysis for planning group technology. *Clarendon Press,* Oxford.

Carmon, T. F., Maimon, O., and Dar-El, E. M., 1989, Group setup for printed circuit board assembly. *International Journal of Production Research*, Vol. 27, No. 10, 1795-1810.

Chandrasekharan, M. P., and Rajagopalan, R., 1987, ZODIAC - an algorithm for concurrent formation of part-families and machine cells. *International Journal of Production Research*, Vol. 25, No. 6, 835-850.

Clark, R. H., 1985, *Handbook of printed circuit manufacturing*, Van Nostrand Reinhold Company - New York.

Cunningham, P., and Browne, J., 1986, A LISP-based heuristic scheduler for automatic insertion in electronics assembly. *International Journal of Production Research*, Vol. 24, No. 6, 1395-1408.

Davis, L., 1991, Handbook of Genetic Algorithms, *Van Nostrand Reinhold,* New York.

De Beer, C., De Witee, J., 1978, Production flow synthesis. *Annals of the CIRP,* 27, 389.

Garey, M.R., Johnson, D. S., 1979, Computers and Intractability: a guide to the theory of NP completeness. *Freeman and Company, New York, NY.*

Goldberg, D. E., 1989, Genetic Algorithms in search, optimization and machine learning, *Addison-Wesley,* Reading, Massachusetts.

Grefenstette, J. J., 1986, Optimization of control parameters for Genetic Algorithms, *IEEE Trans. Systems, Man and Cybernetics,* Vol. SMC-16, No.1, 122-128/

Groover, M. P., 1987, Automation, Production Systems, and Computer Integrated Manufacturing. *Prentice Hall: Englewood Cliffs, Nj.*

Harhalakis, G., Nagi, R., and Proth, J. M., 1990, An efficient heuristic in manufacturing cell formation for group technology applications. *International Journal of Production Research*, Vol. 28, No. 1, 185-198.

Hashiba, S., and Chang, T. C., 1991, Assembly setup reduction using group technology. Proceedings of the 13th Annual Conference on Computers and Industrial Engineering, *Computers and Industrial engineering*, Vol. 21, Nos. 1-4, 453-457.

Hashiba, S., and Chang, T. C., 1992, Heuristic and simulated annealing approaches to PCB assembly setup reduction. Human Aspects in Computer Integrated Manufacturing, *IFIP Transactions, Applications in Technology,* Elsevier Science Publishers.

Holland, J. H., 1975, Adaptation in natural and artificial systems. *University of Michigan Press.*

Huand, Y., Srihari, K., Adriance, J., and Westby, G., 1993, A solution methodology for the multiple batch surface mount PCB placement sequence problem. *Advances in Electronics Packaging – ASME.* EEP Vol. 4, No. 1, 373 – 379.

King, J. R., 1980, Machine-component grouping in production flow analysis: an approach using a rank order clustering algorithm. *International Journal of Production Research*, Vol. 18, 213-232.

King, J. R., and Nakornchai, V., 1984, Machine component group formation in group technology: review and extension. *International Journal of Production Research*, Vol. 20, 117-133.

Kirkpatrick, C. D., 1983, Optimization by simulated annealing, *Science,* Vol.220, No.4598, 671-681.

Koza, J. R., 1994, Genetic Programming II: automatic discovery of reusable programs. *MIT Press*, Cambridge, Mass.

Kumar, C. S., and Chandrasekharan, M. P., 1990, Grouping efficacy: a quantitative orientation for goodness of block diagonal forms of binary matrices in group technology. *International Journal of Production Research*, Vol. 28, No. 2, 233-243.

Kusiak, A., and Chow, W. S., 1988, Decomposition of manufacturing systems. *IEEE Journal of Robotics and Automation*, Vol. 4, No. 5.

Kusiak, A., Heragu, S. S., 1987, Group Technology. *Computers in Industry,* Vol. 9, 83-91.

Lawrence, D., 1991, Handbook of genetic algorithms. *Van Nostrand Reinhold.*

Logendran, R., 1990, A workload based model for minimizing total intercell and intracell moves in cellular manufacturing. *International Journal of Production Research,* Vol. 28, 913-925.

Logendran, R., 1991, Impact of sequence of operations and layout of cells in cellular manufacturing. *International Journal of Production Research*, Vol. 29, 375-390.

Logendran, R., 1992, A model for duplicating bottleneck machines in the presence of budgetary limitations in cellular manufacturing. *International Journal of Production Research,* Vol. 30, 683-694.

Logendran, R., Ramakrishna, P., Sriskandarajah, C., 1993, Tabu search based heuristics for cellular manufacturing systems in the presence of alternative process plans. *International Journal of Production Research,* Vol.32, No. 2, 273-297.

Luzzatto, D., and Perona, M., 1993, Cell formation in PCB assembly based on production quantitative data. *European Journal of Operational Research*, Vol. 69, 312-329.

McAuley, J., 1972, Machine grouping for efficient production. *The production engineer,* 51-53.

Mainmon, O., and Shtub, A., 1991, Grouping methods for printed circuit board assembly. *International Journal of Production Research*, Vol. 29, No. 7, 1379-1390.

Matsushita Electric Industrial Co. Ltd., 1995, *Panasert - MVII Technical Guide.*

Michalewicz, Z., 1992, Genetic Algorithms + data structures = evolutionary programs, *Springler-Verlag,* Berlin.

Mitrofanov, S. P., 1966, The scientific principles of group technology. *Leningrad, 1959.* Translated by the national lending library.

Peters, B. A., and Subramanian, G. S., 1996, Analysis of partial setup strategies for solving the operational planning problem in parallel machine electronic assembly systems. *International Journal of Production Research*, Vol. 4, 999-1021.

Sadiq, M., and Landers, T. L., 1992, Decision support system for intelligent part/slot assignment on a SMT placement machine. *Proceeding of the Technical Program, NEPCON West.*

Sadiq, M., Landers, T. L., and Taylor, G. D., 1993, A heuristic algorithm for minimizing total reduction time for a sequence of jobs on a surface mount placement machine. *International Journal of Production Research*, Vol. 31, No. 6, 1327-1341.

Schwefel, H. P., 1981, Numerical optimization of computer models, *Wiley,* Chichester, UK.

Seifoddini, H., and Djassemi, M., 1996, A new grouping measure for evaluation of machine-component matrices. *International Journal of Production Research*, Vol. 34., No. 5, 1179-1193.

Shiko, G., 1992, A process planning-oriented approach to part family formation problem in group technology applications. *International Journal of Production Research*, Vol. 30, No. 8, 1739-1752.

Shtub, A., and Maimon, O., 1992, Role of similarity measures in PCB grouping procedures. *International Journal of Production Research*, Vol. 30, No. 5, 973-983.

Sokal, R. R., and Sneath, P. H. A., 1963, Principles of numerical taxonomy. *Freeman and Company*, San Francisco, Ca.

Srinivas, M., and Patnaik, L. M., 1994, Genetic Algorithms: a survey. *Computer*, Vol. 27, No. 6, 17-26.

Stecke, K. E., and Kim, I., 1991, A flexible approach to part type selection in flexible flow systems using part mix ratios. *International Journal of Production Research*, Vol. 1991, No. 1, 53-75.

Tang, C. S., Dernardo, E. V., 1988, Models arising from a flexible manufacturing machine part I: Minimization of the number of tool switches. *Operations Research*, Vol.36.

Ventura, J. A., Chen, F. F., and Wu, C. H., 1990, Grouping parts and tools in flexible manufacturing systems production planning. *International Journal of Production Research*, Vol. 28, No. 6, 1039-1056.

APPENDICES

APPENDIX A

Experim. Run	Factors						Results		
	GA Runs	Popsize	BestRepet	MutElem	PC	PM	TotCost ($)	CompTime (sec.)	Configuration
1	50	50	10	6	0.6	0.1	6607	150.17	2,2,2,2,2,2,0,2,1,2,2,2,0,2,2,2,2,1,2,2
2	30	50	6	6	0.6	0.1	6784	91.12	2,2,2,2,2,2,2,2,2,2,2,2,0,2,1,2,2,0,2
3	50	100	10	6	0.6	0.2	6560	326.37	2,2,2,2,2,2,1,0,2,2,2,2,2,2,2,0,2,2,0,2
4	50	100	6	6	0.6	0.1	7879	286.77	2,2,0,2,2,0,2,2,1,2,0,0,2,2,1,2,0,2,2,1
5	30	100	10	10	0.8	0.1	6932	189.39	2,2,1,2,2,2,2,2,2,2,0,0,2,2,2,0,2,1,0,2
6	50	100	6	10	0.8	0.1	8486	313.07	1,2,1,2,2,1,1,2,2,2,2,2,0,2,2,0,0,2,0,2
7	30	50	10	6	0.6	0.2	7571	97.44	2,2,0,2,2,2,2,1,1,2,0,0,0,2,1,2,0,2,2,2
8	30	100	6	6	0.8	0.1	6538	186.8	2,2,1,2,2,0,1,1,2,2,1,2,2,2,2,2,2,1,2,2
9	30	50	6	6	0.8	0.2	6982	100.4	2,2,1,2,2,2,2,1,2,2,1,2,0,2,2,2,2,1,2,1
10	50	100	6	6	0.8	0.2	6377	325	2,2,0,2,2,0,2,0,2,2,2,2,2,2,2,2,2,1,2,2
11	30	100	10	6	0.6	0.1	8124	185.37	2,0,2,2,2,0,2,2,1,0,2,2,0,2,2,2,0,2,0,2
12	50	50	10	10	0.6	0.2	9362	161.92	2,2,0,2,1,0,1,1,2,2,1,2,2,2,2,0,2,1,2,0
13	50	50	6	6	0.8	0.1	8433	122.09	2,2,0,0,2,1,2,2,1,2,0,2,2,0,2,2,2,1,0,1
14	30	50	6	10	0.8	0.1	8666	90.3	2,1,0,2,2,2,2,2,1,2,0,1,2,2,2,2,0,2,0,1
15	50	50	10	10	0.8	0.1	8855	150.05	2,2,0,1,2,1,2,0,2,0,1,2,0,2,2,0,2,1,2,2
16	50	100	10	10	0.8	0.2	7701	331.15	2,2,2,0,2,2,0,2,2,2,0,2,2,0,2,2,2,1,0,2
17	50	100	10	6	0.8	0.1	7078	310.61	2,2,1,2,2,2,2,2,2,0,2,2,0,2,1,2,2,2,2,1
18	50	100	10	10	0.6	0.1	8517	311.04	2,2,1,2,1,2,2,0,2,0,2,1,2,2,2,2,0,1,0,2
19	30	100	6	10	0.8	0.2	7596	201.41	1,2,0,2,2,0,1,2,2,2,1,2,0,0,2,2,0,2,2,2
20	50	50	6	6	0.6	0.2	7459	158.96	2,2,1,2,2,2,1,1,2,2,2,2,0,2,2,0,2,1,2,1
21	30	100	6	10	0.6	0.1	6934	190.21	2,2,2,2,2,2,2,2,1,0,1,2,2,2,1,2,0,1,0,2
22	30	100	10	6	0.8	0.2	7270	195.92	2,2,2,2,2,2,1,1,2,1,2,2,2,0,2,2,2,1,2,2
23	50	50	6	10	0.8	0.2	9188	160.72	2,2,1,1,2,1,2,0,2,0,2,0,2,2,2,0,2,2,2,2
24	50	50	6	10	0.6	0.1	7564	149.95	2,2,2,2,2,2,2,1,2,0,1,2,0,2,1,2,2,1,2,1
25	30	50	10	6	0.8	0.1	9393	92.6	2,2,1,2,1,0,2,2,1,0,1,1,0,1,2,0,2,2,2,1
26	30	50	6	10	0.6	0.2	7629	99.36	2,2,0,2,2,0,2,1,2,1,2,1,2,0,2,2,2,2,0,2
27	30	50	10	10	0.8	0.2	8231	99.96	2,2,0,2,2,0,2,2,2,2,2,2,0,0,1,0,2,2,0,0
28	50	50	10	6	0.8	0.2	6270	163.79	2,2,0,2,2,0,2,2,2,0,2,2,2,2,2,2,2,0,2
29	30	50	10	10	0.6	0.1	9548	92.11	2,2,1,0,2,2,1,2,2,2,0,2,0,2,2,0,1,2,2,0
30	50	100	6	10	0.6	0.2	7512	328.23	2,2,1,2,2,1,1,2,2,0,0,2,2,2,1,2,2,2,0,2
31	30	100	10	10	0.6	0.2	7619	200.75	2,2,0,2,2,2,1,2,2,2,0,2,0,2,2,2,2,2,0,0
32	30	100	6	6	0.6	0.2	6689	197.24	2,2,2,2,2,0,2,2,1,0,0,2,0,2,2,2,2,2,2,1

Table A1 - Experimental runs and results Heuristic 1 - Screening Test

Experim. Run	Factors						Results		
	GA Runs	Popsize	BestRepet	MutElem	PC	PM	TotCost ($)	CompTime (sec.)	Configuration
1	50	50	10	6	0.6	0.1	7784	101.39	2,2,1,2,2,0,2,2,1,2,1,0,2,2,0,2,0,1,2,2
2	30	50	6	6	0.6	0.1	7002	61.95	2,2,0,0,2,0,2,2,1,2,0,2,2,2,2,2,0,2,2,2
3	50	100	10	6	0.6	0.2	5901	229.15	2,2,2,2,2,0,2,2,1,2,2,2,2,2,2,2,1,2,2
4	50	100	6	6	0.6	0.1	8276	209.38	2,2,2,2,2,0,2,0,2,0,2,0,2,2,2,1,0,2,2,2
5	30	100	10	10	0.8	0.1	8459	129.79	2,2,2,2,2,1,1,1,2,1,2,2,2,0,1,0,2,2,0,2
6	50	100	6	10	0.8	0.1	8299	211.24	2,2,0,0,2,0,2,2,1,0,1,1,0,1,2,2,2,2,2,2
7	30	50	10	6	0.6	0.2	8893	66.08	2,1,1,2,2,2,2,1,2,1,2,0,0,0,2,0,2,1,0,2
8	30	100	6	6	0.8	0.1	7855	128.97	2,0,2,2,2,2,2,2,1,0,2,2,2,0,2,2,2,1,0,2
9	30	50	6	6	0.8	0.2	7371	68.71	2,2,1,2,2,0,2,2,1,2,0,2,0,1,2,2,0,2,2,1
10	50	100	6	6	0.8	0.2	5869	229.37	2,2,1,2,2,2,2,2,2,2,1,2,2,2,2,2,2,2,2,2
11	30	100	10	6	0.6	0.1	8694	128.08	2,2,2,1,2,2,2,1,1,0,1,2,2,0,2,0,2,2,0,1
12	50	50	10	10	0.6	0.2	7883	108.53	2,2,0,2,2,1,1,2,2,0,0,2,0,2,2,2,2,1,2,1
13	50	50	6	6	0.8	0.1	6376	101.28	2,2,2,2,2,0,2,0,1,2,2,2,2,0,2,2,2,1,0,2
14	30	50	6	10	0.8	0.1	7845	60.74	2,2,1,2,2,2,0,2,1,0,1,2,2,2,0,2,2,1,2,2
15	50	50	10	10	0.8	0.1	9576	101.01	2,2,2,0,1,2,2,2,1,2,2,0,0,1,2,2,2,1,2,1
16	50	100	10	10	0.8	0.2	7633	231.51	2,2,2,0,2,2,2,2,1,0,0,1,2,0,2,2,2,2,2,2
17	50	100	10	6	0.8	0.1	6771	214.37	2,2,0,2,2,2,2,0,2,2,1,2,0,2,1,2,2,1,0,2
18	50	100	10	10	0.6	0.1	7318	212.34	2,2,2,2,2,2,2,1,1,0,2,2,0,1,2,0,2,1,2,2
19	30	100	6	10	0.8	0.2	7996	140.39	2,2,2,2,1,2,2,0,2,2,2,2,2,2,0,2,2,0,1
20	50	50	6	6	0.6	0.2	6652	108.42	2,2,0,2,2,2,1,2,2,0,1,2,2,2,2,0,2,2,2,2
21	30	100	6	10	0.6	0.1	7749	129.73	2,2,0,2,2,2,1,2,1,0,0,2,2,2,1,1,2,2,0,2
22	30	100	10	6	0.8	0.2	6911	141.17	2,2,1,2,2,0,1,2,1,2,1,2,2,0,2,2,2,2,0,1
23	50	50	6	10	0.8	0.2	9982	135.91	2,0,2,0,2,2,2,0,1,2,2,2,0,1,1,2,2,2,0,1
24	50	50	6	10	0.6	0.1	8667	67.34	2,2,0,1,2,2,1,2,1,2,2,0,2,0,1,0,2,2,2,2
25	30	50	10	6	0.8	0.1	8866	42.79	2,2,1,0,1,2,2,2,2,0,0,2,2,0,2,2,2,2,2,1
26	30	50	6	10	0.6	0.2	7103	45.92	2,1,1,2,2,2,1,2,2,2,2,2,2,2,2,2,2,2,2,2
27	30	50	10	10	0.8	0.2	6356	47.29	2,2,2,2,2,2,1,2,1,0,2,2,2,2,2,2,2,1,2,2
28	50	50	10	6	0.8	0.2	7838	135.13	2,2,2,2,2,2,1,2,2,0,2,1,0,2,2,0,2,2,2,1
29	30	50	10	10	0.6	0.1	7318	42.73	2,2,2,2,2,2,1,2,1,0,0,2,0,1,2,0,2,1,2,2
30	50	100	6	10	0.6	0.2	8531	153.84	2,2,0,1,2,1,2,2,1,0,2,2,2,2,1,2,2,1,0,2
31	30	100	10	10	0.6	0.2	7123	92.49	2,2,1,2,2,1,1,2,2,2,2,2,2,2,1,2,2,2,2,2
32	30	100	6	6	0.6	0.2	5837	93.92	2,2,2,2,2,0,2,2,2,2,0,2,2,2,2,2,2,2,2,2

Table A2 - Experimental runs and results Heuristic 2 - Screening Test

Experim. Run	Factors						Results		
	GA Runs	Popsize	BestRepet	MutElem	PC	PM	TotCost ($)	CompTime (sec.)	Configuration
1	50	50	10	6	0.6	0.1	10332	59.76	1,2,0,0,2,0,0,0,1,2,2,2,0,0,1,0,2,1,0,1
2	30	50	6	6	0.6	0.1	9445	37.02	2,2,1,0,2,1,2,2,1,1,1,2,0,0,0,2,0,1,2,2
3	50	100	10	6	0.6	0.2	9071	186.25	2,2,2,2,1,2,2,2,1,2,0,1,0,0,1,2,1,1,2,2
4	50	100	6	6	0.6	0.1	9837	135.12	2,2,2,1,1,2,1,0,2,2,1,0,2,2,1,2,0,2,2,2
5	30	100	10	10	0.8	0.1	8326	81.13	2,2,1,2,2,2,2,1,2,2,0,0,2,1,2,2,2,1,2,1
6	50	100	6	10	0.8	0.1	9243	133.25	2,2,2,0,1,0,1,2,2,1,1,2,0,2,1,0,2,1,2,2
7	30	50	10	6	0.6	0.2	9959	46.74	1,2,2,2,1,2,1,1,2,2,2,2,2,0,0,0,0,2,0,2
8	30	100	6	6	0.8	0.1	9248	79.2	2,2,0,2,2,2,1,2,1,2,1,2,0,2,1,2,2,0,2,0
9	30	50	6	6	0.8	0.2	8760	45.53	2,2,2,0,2,1,0,0,1,2,2,2,2,2,1,2,2,1,2,2
10	50	100	6	6	0.8	0.2	7634	169.06	2,2,1,2,1,2,2,2,2,0,2,2,0,2,2,2,0,1,0,2
11	30	100	10	6	0.6	0.1	6409	82.67	2,2,2,2,2,0,2,2,2,2,0,2,0,0,2,2,0,2,0,2
12	50	50	10	10	0.6	0.2	8298	78.27	2,2,2,2,1,0,1,2,2,2,2,2,2,0,1,2,2,1,0,1
13	50	50	6	6	0.8	0.1	10619	59.05	1,2,0,2,1,0,1,2,1,0,2,2,2,1,1,0,1,2,2,2
14	30	50	6	10	0.8	0.1	9618	36.3	2,2,2,2,2,1,1,2,0,0,1,2,0,2,1,2,1,2,2,2
15	50	50	10	10	0.8	0.1	7974	58.38	2,2,1,2,2,0,1,0,2,0,1,2,2,2,0,0,0,2,2,2
16	50	100	10	10	0.8	0.2	9514	178.95	2,2,1,2,0,2,1,2,2,0,1,0,0,2,1,2,2,1,2,1
17	50	100	10	6	0.8	0.1	7713	128.96	2,2,1,1,2,1,2,2,2,2,0,2,0,2,2,2,2,2,2,2
18	50	100	10	10	0.6	0.1	8431	133.14	2,2,0,0,2,2,1,0,2,2,1,2,0,0,2,1,2,1,2,2
19	30	100	6	10	0.8	0.2	8315	112.26	2,1,1,1,2,2,2,2,1,2,2,2,2,2,2,0,2,0,2,2
20	50	50	6	6	0.6	0.2	8166	75.9	2,0,1,0,2,2,2,2,2,2,2,2,2,0,2,2,2,2,2,2
21	30	100	6	10	0.6	0.1	8458	80.14	2,2,0,2,1,2,2,2,1,2,0,0,2,1,1,2,2,2,2,2
22	30	100	10	6	0.8	0.2	9037	103.59	2,0,1,2,2,1,2,1,2,2,2,2,2,0,2,2,0,1,0,2
23	50	50	6	10	0.8	0.2	8435	76.34	1,2,2,2,2,0,1,1,2,0,1,2,2,0,2,0,0,2,2,2
24	50	50	6	10	0.6	0.1	6909	59.65	2,2,2,2,2,2,2,0,2,0,2,2,2,2,2,0,0,2,0,2
25	30	50	10	6	0.8	0.1	10401	37.24	2,2,1,2,0,2,2,1,2,1,2,0,2,2,0,0,0,1,0,2
26	30	50	6	10	0.6	0.2	8357	48.8	2,2,2,2,2,2,2,2,2,2,2,2,2,0,0,2,0,2,2,0
27	30	50	10	10	0.8	0.2	10913	46.03	2,0,2,2,1,0,2,1,2,1,2,2,2,2,1,0,0,2,0,1
28	50	50	10	6	0.8	0.2	10499	72.38	2,2,0,2,0,1,2,2,2,0,0,0,2,0,2,2,2,0,0,2
29	30	50	10	10	0.6	0.1	10627	40.81	2,2,0,2,2,1,1,0,1,2,2,2,1,2,1,0,0,2,2,0
30	50	100	6	10	0.6	0.2	8161	182.79	2,2,1,0,2,2,2,2,1,2,1,1,2,0,0,2,2,2,0,2
31	30	100	10	10	0.6	0.2	8858	115.02	2,2,0,2,2,1,1,1,1,0,1,2,0,1,1,2,0,2,2,2
32	30	100	6	6	0.6	0.2	8357	112.11	2,2,1,2,1,2,2,0,1,2,2,1,2,2,2,2,2,2,2,0,1

Table A3 - Experimental runs and results Heuristic 3 - Screening Test

Table A4 - Analysis of Variance for TotCost - Heuristic 1 - Screening Test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	56280.1	1	56280.1	0.08	0.7856
B:Popsize	3.5979E+06	1	3.5979E+06	4.99	0.0495
C:BestRepet	757065.0	1	757065.0	1.05	0.3296
D:MutElem	6.41357E+06	1	6.41357E+06	8.90	0.0137
E:PC	83845.1	1	83845.1	0.12	0.7401
F:PM	1.24899E+06	1	1.24899E+06	1.73	0.2175
AB	3777146.0	1	377146.0	0.52	0.4861
AC	2.42991E+06	1	2.42991E+06	3.37	0.0963
AD	1.41036E+05	1	1.41036E+06	1.96	0.1922
AE	1485.13	1	1485.13	0.00	0.9647
AF	3655.13	1	3655.13	0.01	0.9446
BC	56280.1	1	56280.1	0.08	0.7856
BD	708645.0	1	708645.0	0.98	0.3449
BE	894453.0	1	894453.0	1.24	0.2914
BF	1.125	1	1.125	0.00	0.999
CD	66430.1	1	66430.1	0.09	0.7677
CE	1.12275E+06	1	1.12275E+06	1.56	0.2405
CF	214185.0	1	214185.0	0.30	0.5977
DE	2850.13	1	2850.13	0.00	0.9511
DF	779376.0	1	779376.0	1.08	0.323
EF	322003.0	1	322003.0	0.45	0.5191
Total Error	7.20979E+06	10	720979.0		
Total	2.7757E+07	31			
R-squared	74.0253%				

Table A5 - Analysis of Variance for TotCost - Heuristic 1 - Screening Test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
B:Popsize	3.5979E+06	1	3.5979E+06	5.88	0.0218
D:MutElem	6.41357E+06	1	6.41357E+06	10.48	0.0030
Total Error	7.20979E+06	29	611914.0		
Total	2.7757E+07	31			
R-squared	36.0683%				

Table A6 - Analysis of Variance for CompTime - Heuristic 1 - Screening Test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	64755.9	1	64755.9	1912.80	0.0000
B:Popsize	137601.0	1	137601.0	4064.54	0.0000
C:BestRepet	101.567	1	101.567	3.00	0.1139
D:MutElem	194.883	1	194.883	5.76	0.0374
E:PC	1.2207	1	1.2207	0.04	0.8532
F:PM	1754.84	1	1754.84	51.84	0.0000
AB	8804.31	1	8804.31	260.07	0.0000
AC	126.445	1	126.445	3.73	0.0821
AD	65.4654	1	65.4654	1.93	0.1945
AE	0.000378125	1	0.000378125	0.00	0.9974
AF	240.956	1	240.956	7.12	0.0236
BC	5.5029	1	5.5029	0.16	0.6953
BD	17.1	1	17.1	0.50	0.4939
BE	734775.0	1	734775.0	2.17	0.1715
BF	25.6507	1	25.6507	0.76	0.4045
CD	80.5498	1	80.5498	2.38	0.1540
CE	3.34758	1	3.34758	0.10	0.7596
CF	63.4	1	63.4	1.87	0.2010
DE	0.0520031	1	0.0520031	0.00	0.9695
DF	55.7	1	55.7	1.64	0.2286
EF	3.1	1	3.1	0.09	0.7695
Total Error	338.541	10	33.8541		
Total	214313.0	31			
R-squared	99.8420%				

Table A7 - Analysis of Variance for CompTime - Heuristic 1 - Screening Test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	64755.9	1	64755.9	1684.50	0.0000
B:Popsize	137601.0	1	137601.0	3579.43	0.0000
D:MutElem	194.883	1	194.883	5.07	0.0334
F:PM	1754.84	1	1754.84	45.65	0.0000
AB	8804.31	1	8804.31	229.03	0.0000
AF	240.956	1	240.956	6.27	0.0192
Total Error	338.541	25	38.4422		
Total	214313.0	31			
R-squared	99.5516%				

Table A8 - Analysis of Variance for TotCost - Heuristic 2 - Screening Test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	122265.0	1	122265.0	0.14	0.7125
B:Popsize	1.2364E+06	1	1.2364E+06	1.45	0.2556
C:BestRepet	114481.0	1	114481.0	0.13	0.7213
D:MutElem	3.74148E+06	1	3.74148E+06	4.40	0.0623
E:PC	334562.0	1	334562.0	0.39	0.5446
F:PM	2.51777E+06	1	2.51777E+06	2.96	0.1160
AB	1136280.0	1	1136280.0	1.34	0.2746
AC	1.05488E+06	1	1.05488E+06	1.24	0.2914
AD	6.03955E+06	1	6.03955E+06	7.10	0.0237
AE	11552.0	1	11552.0	0.01	0.9095
AF	365513.0	1	365513.0	0.43	0.5269
BC	818560.0	1	818560.0	0.96	0.3497
BD	289941.0	1	289941.0	0.34	0.5722
BE	202248.0	1	202248.0	0.24	0.6363
BF	1.22618E+06	1	1.23E+06	1.44	0.2575
CD	3.73055E+06	1	3.73E+06	4.39	0.0627
CE	2.45000E+03	1	2.45000E+03	0.00	0.9583
CF	387200.0	1	387200.0	0.46	0.5151
DE	992641.0	1	992641.0	1.17	0.3053
DF	434312.0	1	434312.0	0.51	0.4912
EF	19701.1	1	19701.1	0.02	0.8820
Total Error	8.50352E+06	10	8.50352E+06		
Total	3.3282E+07	31			
R-squared	74.4501%				

Table A9 - Analysis of Variance for TotCost - Heuristic 2 - Screening Test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
D:MutElem	3.74148E+06	1	3.74148E+06	5.86	0.0225
F:PM	2.51777E+06	1	2.51777E+06	3.94	0.0574
AD	6.03955E+06	1	6.03955E+06	9.45	0.0048
CD	3.73055E+06	1	3.73E+06	2.96	0.1227
Total Error	8.50352E+06	27	8.50352E+06		
Total	3.3282E+07	31			
R-squared	48.1622%				

Table A10 - Analysis of Variance for CompTime - Heuristic 2 - Screening Test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	39865.0	1	39865.0	254.25	0
B:Popsize	59557.4	1	59557.4	379.85	0
C:BestRepet	184.0	1	184.0	1.17	0.3041
D:MutElem	697.138	1	697.138	4.45	0.0612
E:PC	2250.9	1	2250.9	14.36	0.0035
F:PM	224.19	1	224.19	1.43	0.2594
AB	2518.0	1	2518.0	16.06	0.0025
AC	765.97	1	765.97	4.89	0.0515
AD	128.721	1	128.721	0.82	0.3862
AE	155.2	1	155.2	0.99	0.3432
AF	633.0	1	633.0	4.04	0.0723
BC	238.6	1	238.6	1.52	0.2456
BD	0.32	1	0.32	0.00	0.9649
BE	238.6	1	238.6	1.52	0.2456
BF	1114.16	1	1114.16	7.11	0.0237
CD	39.5605	1	39.5605	0.25	0.6263
CE	646.561	1	646.561	4.12	0.0697
CF	166.5	1	166.5	1.06	0.327
DE	626.0	1	626.0	3.99	0.0736
DF	214.1	1	214.1	1.37	0.2696
EF	1174.7	1	1174.7	7.49	0.0209
Total Error	1567.93	10	156.793		
Total	113007.0	31			
R-squared	98.6125%				

Table A11 - Analysis of Variance for CompTime - Heuristic 2 - Screening Test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	39865.0	1	39865.0	152.70	0.0000
B:Popsize	59557.4	1	59557.4	228.14	0.0000
E:PC	2250.9	1	2250.9	8.62	0.007
AB	2518.0	1	2518.0	9.65	0.0047
BF	1114.16	1	1114.16	4.27	0.0494
EF	1174.7	1	1174.7	4.50	0.0440
Total Error	6526.55	25	261.062		
Total	113007.0	31			
R-squared	94.2200%				

Table A12 - Analysis of Variance for TotCost - Heuristic 3 - Screening Test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	564985.0	1	564985.0	0.59	0.4587
B:Popsize	5.04031E+06	1	5.0403E+06	5.30	0.0441
C:BestRepet	1445000.0	1	1445000.0	1.52	0.2459
D:MutElem	796953.0	1	796953.0	0.84	0.3815
E:PC	1350550.0	1	1350550.0	1.42	0.2609
F:PM	49298.0	1	49298.0	0.05	0.8245
AB	2787160.0	1	2787160.0	2.93	0.1177
AC	40898.0	1	40898.0	0.04	0.8399
AD	2.39915E+06	1	2.39915E+06	2.52	0.1433
AE	92665.1	1	92665.1	0.10	0.7613
AF	53138.0	1	53138.0	0.06	0.8179
BC	3503300.0	1	3503300.0	3.68	0.0839
BD	2559450.0	1	2559450.0	2.69	0.1319
BE	422740.0	1	422740.0	0.44	0.5201
BF	456013.0	1	456013.0	0.48	0.5044
CD	522753.0	1	522753.0	0.55	0.4755
CE	100128.0	1	100128.0	0.11	0.7523
CF	5.38576E+06	1	5.38576E+06	5.66	0.0386
DE	113288.0	1	113288.0	0.12	0.7371
DF	447931.0	1	447931.0	0.47	0.5081
EF	43956.1	1	43956.1	0.05	0.8341
Total Error	9.51089E+06	10	951089.0		
Total	3.7686E+07	31			
R-squared	74.7630%				

Table A13 - Analysis of Variance for TotCost - Heuristic 3 - Screening Test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
B:Popsize	5.04031E+06	1	5.04031E+06	7.87	0.0096
AB	2787160.0	1	2787160.0	4.35	0.0473
AD	2.39915E+06	1	2.39915E+06	3.75	0.0643
BC	3503300.0	1	3503300.0	5.47	0.0276
BD	2559450.0	1	2559450.0	4.00	0.0566
CF	5.38576E+06	1	5.38576E+06	8.41	0.0077
Total Error	9.51089E+06	25	951089.0		
Total	3.7686E+07	31			
R-squared	57.5146%				

Table A14 - Analysis of Variance for TotCost - Heuristic 3 - Screening Test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
B:Popsize	5.04031E+06	1	5.04031E+06	6.49	0.0169
AB	2787160.0	1	2787160.0	3.59	0.0689
BC	3503300.0	1	3503300.0	4.51	0.0430
CF	5.38576E+06	1	5.38576E+06	6.93	0.0138
Total Error	9.51089E+06	27	951089.0		
Total	3.7686E+07	31			
R-squared	44.3570%				

Table A15 - Analysis of Variance for CompTime - Heuristic 3 - Screening Test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	14563.3	1	14563.3	1211.54	0.0000
B:Popsize	40288.2	1	40288.2	3351.65	0.0000
C:BestRepet	1.4450	1	1.4450	0.12	0.7360
D:MutElem	29.4145	1	29.4145	2.45	0.1488
E:PC	99.8991	1	99.8991	8.31	0.0163
F:PM	5207.1	1	5207.1	433.19	0.0000
AB	2452.5	1	2452.5	204.02	0.0000
AC	0.29261	1	0.29261	0.02	0.8791
AD	0.13781	1	0.13781	0.01	0.9168
AE	4.8672	1	4.8672	0.40	0.5389
AF	294.395	1	294.395	24.49	0.0006
BC	0.70805	1	0.70805	0.06	0.8131
BD	2.39805	1	2.39805	0.20	0.6646
BE	19.75060	1	19.75060	1.64	0.2288
BF	1308.67	1	1308.67	108.87	0.0000
CD	0.18	1	0.18	0.01	0.9050
CE	7.46911	1	7.46911	0.62	0.4488
CF	0.13520	1	0.13520	0.01	0.9176
DE	18.8805	1	18.8805	1.57	0.2386
DF	16.7042	1	16.7042	1.39	0.2658
EF	22.6801	1	22.6801	1.89	0.1996
Total Error	120.204	10	120.204		
Total	64459.3	31			
R-squared	99.8135%				

Table A16 - Analysis of Variance for CompTime - Heuristic 3 - Screening Test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	14563.3	1	14563.3	1425.05	0.0000
B:Popsize	40288.2	1	40288.2	3942.30	0.0000
E:PC	99.8991	1	99.8991	9.78	0.0046
F:PM	5207.1	1	5207.1	509.53	0.0000
AB	2452.5	1	2452.5	239.95	0.0000
AF	294.395	1	294.395	28.81	0.0000
BF	1308.67	1	1308.67	128.06	0.0000
Total Error	120.204	24	8.781		
Total	64459.3	31			
R-squared	99.6195%				

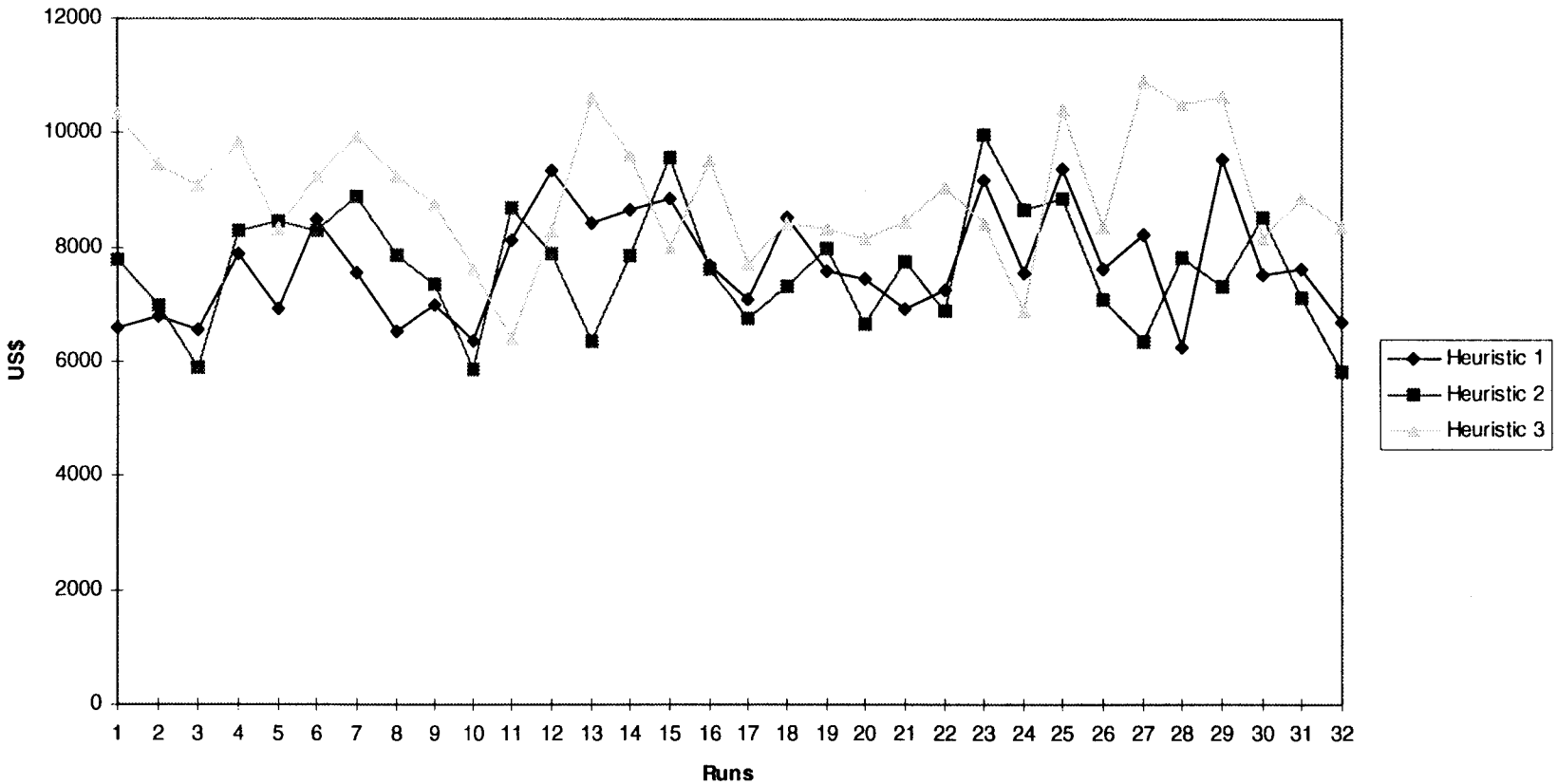


Figure A1 - Results for TotCost - Screening Test

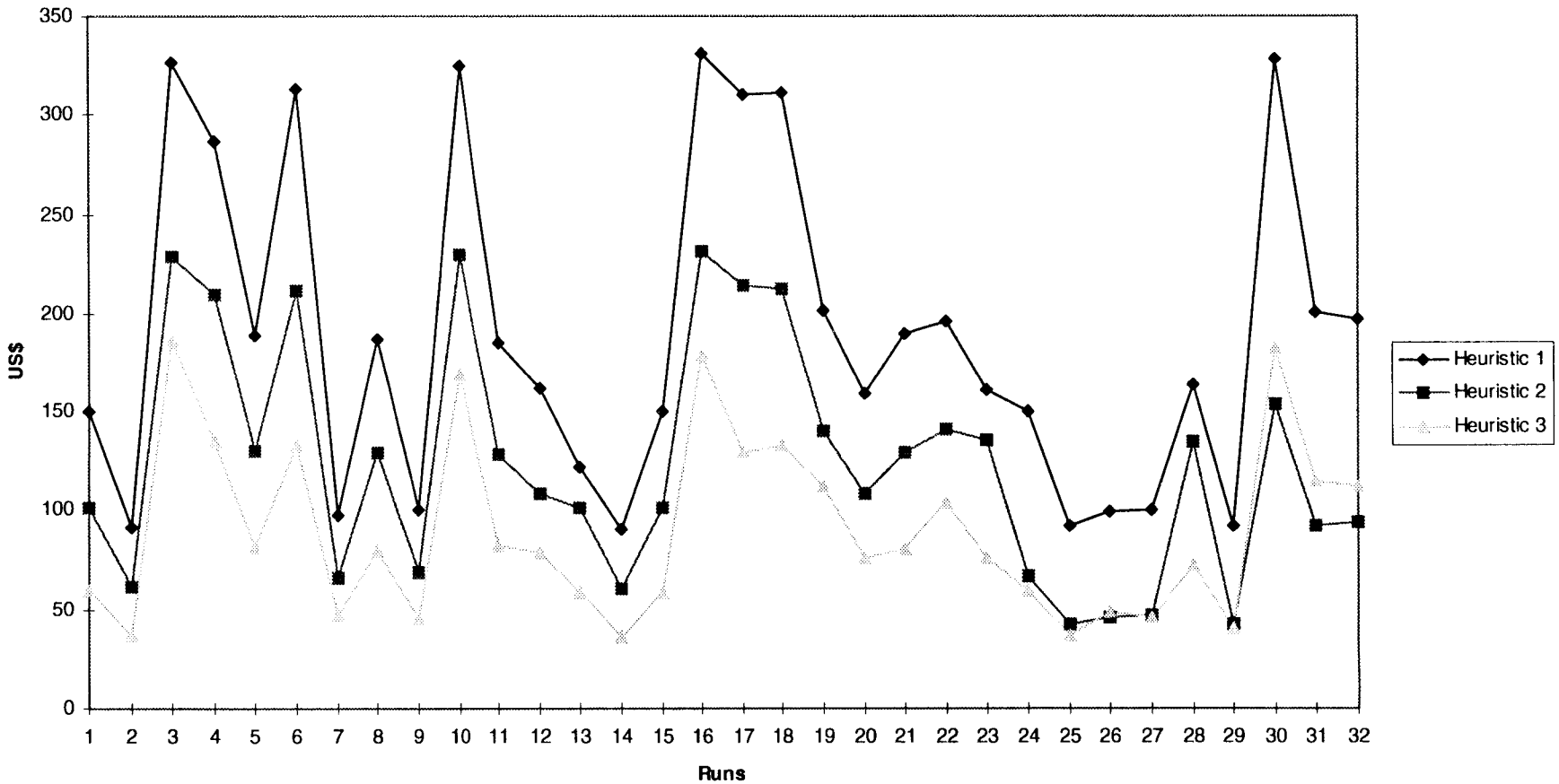


Figure A2 - Results for CompTime - Screening Test

Figure A3 - Normal Probability Plot of Residuals for TotCost Heuristic 1

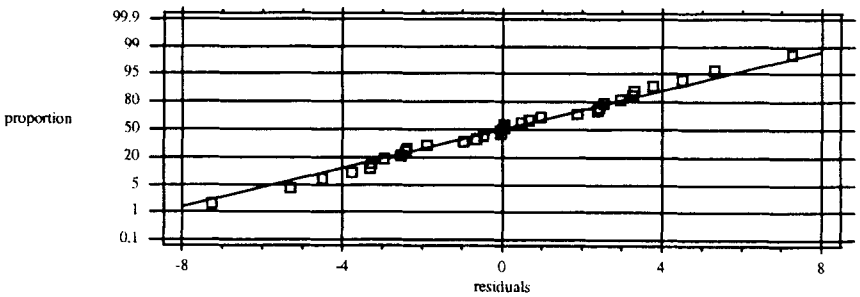


Figure A4 -Residuals Plot for TotCost Heuristic 1

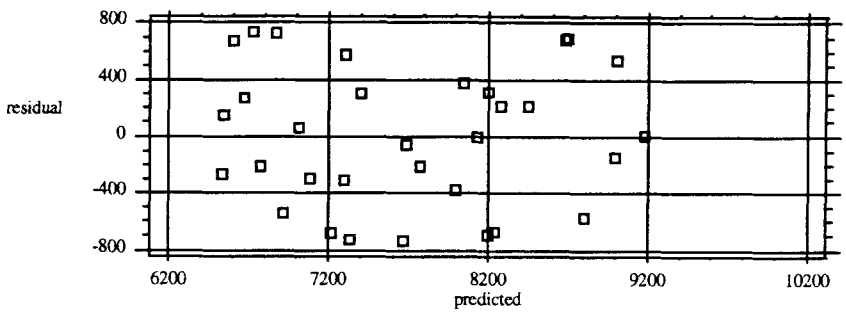


Figure A5 -Main Effects Plot for TotCost Heuristic 1

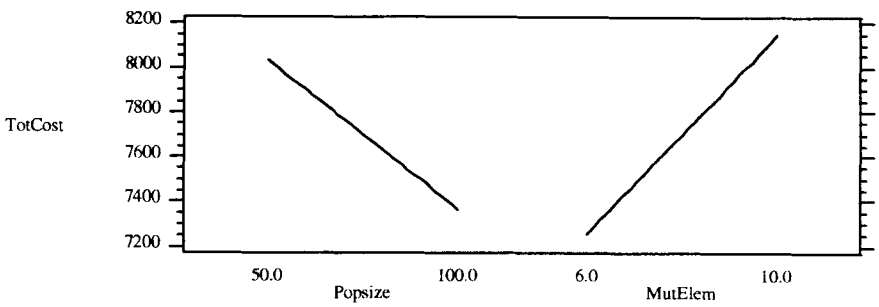


Figure A6 -Normal Probability Plot for CompTime Heuristic 1

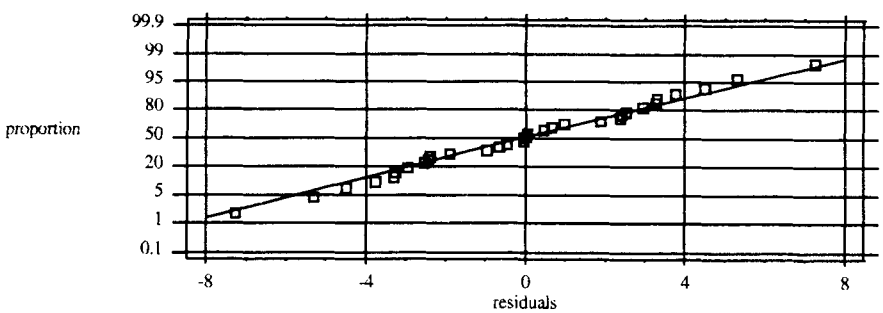


Figure A7 -Residual Plot for CompTime Heuristic 1

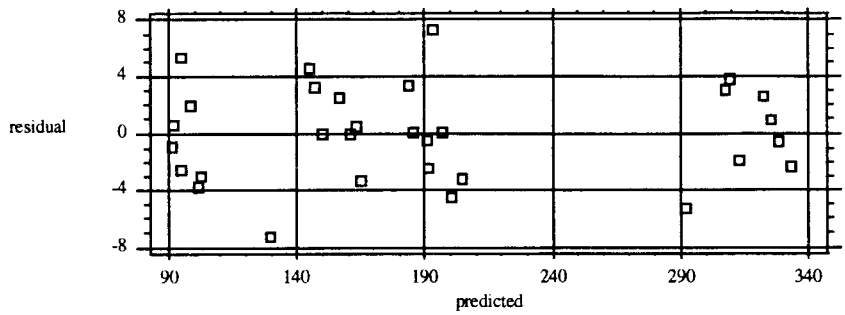


Figure A8 -Main Effects Plot for CompTime Heuristic 1

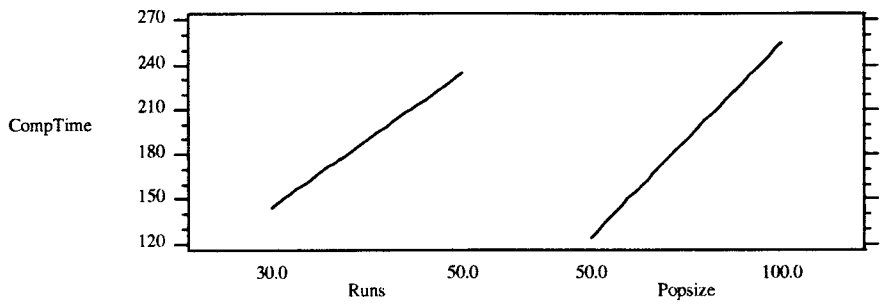


Figure A9 - Main Effects Plot for CompTime Heuristic 1

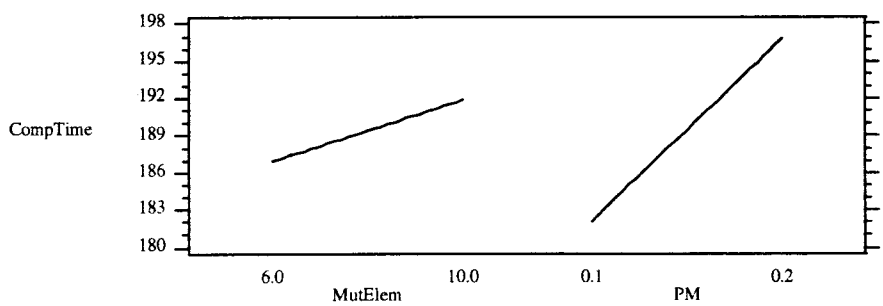


Figure A10 - Interaction Plot for CompTime Heuristic 1

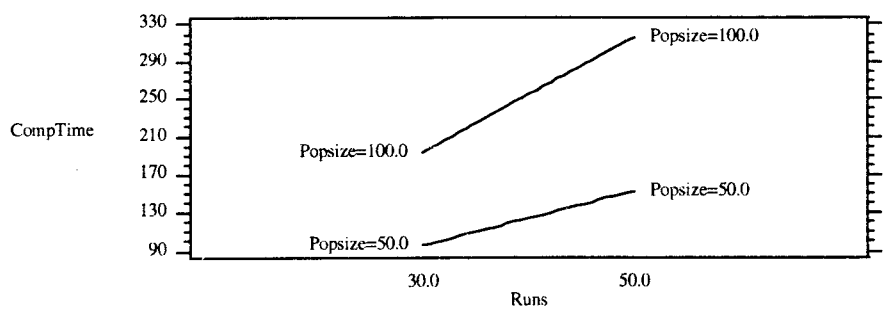


Figure A11 - Interaction Plot for CompTime Heuristic 1

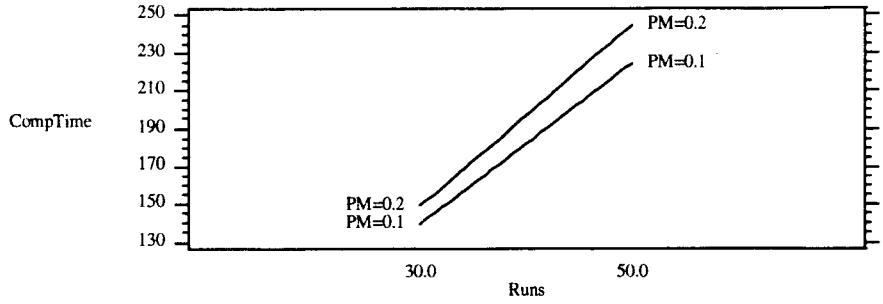


Figure A12 -Normal Probability Plot for Residuals - TotCost Heuristic 2

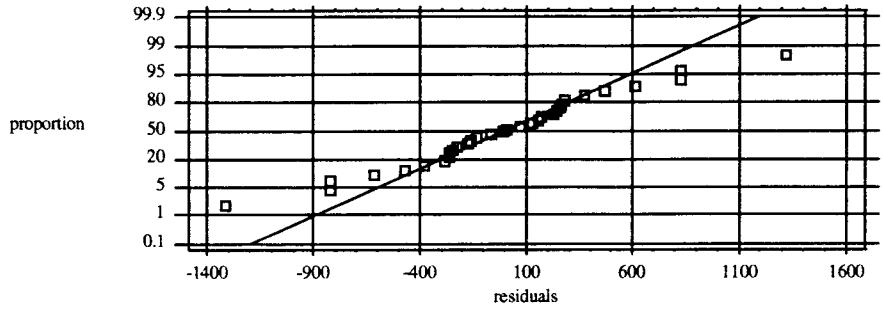


Figure A13 - Residual Plot for TotCost Heuristic 2

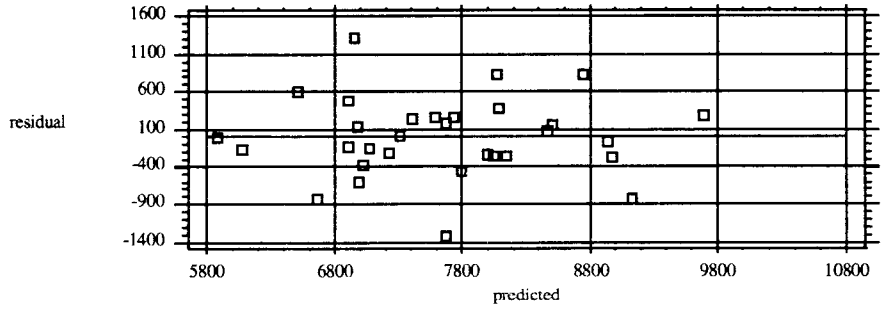


Figure A14 - Main Effects Plot for TotCost Heuristic 2

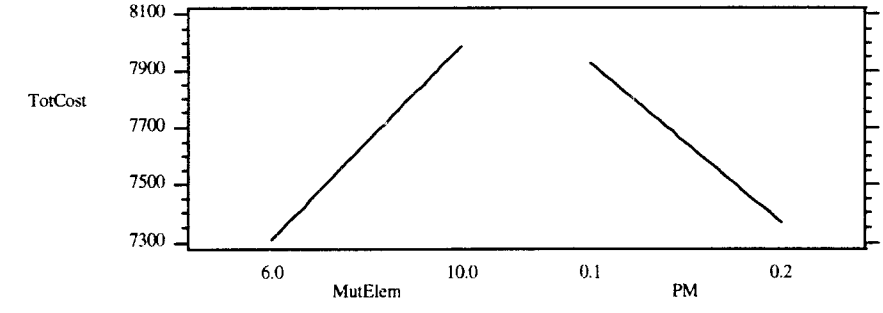


Figure A15 - Interaction Plot for TotCost

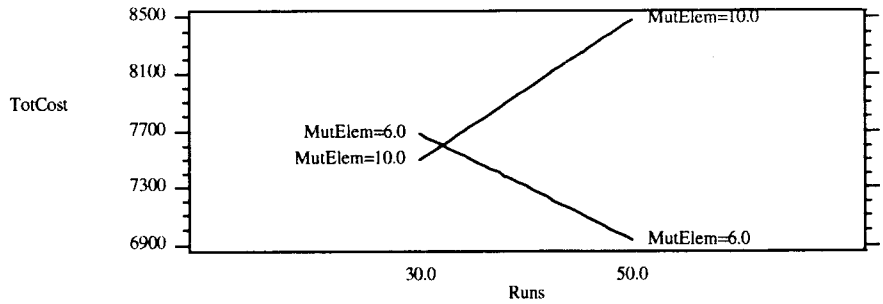


Figure A16 - Normal Probability Plot for Residuals - CompTime Heuristic 2

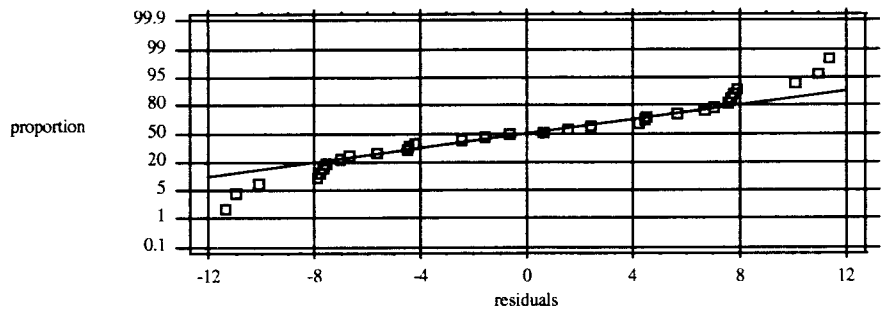


Figure A17 - Residual Plot for CompTime Heuristic 2

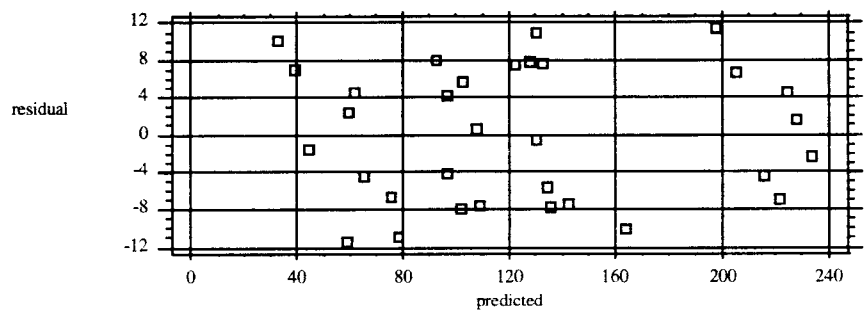


Figure A18 - Main Effects Plot for CompTime

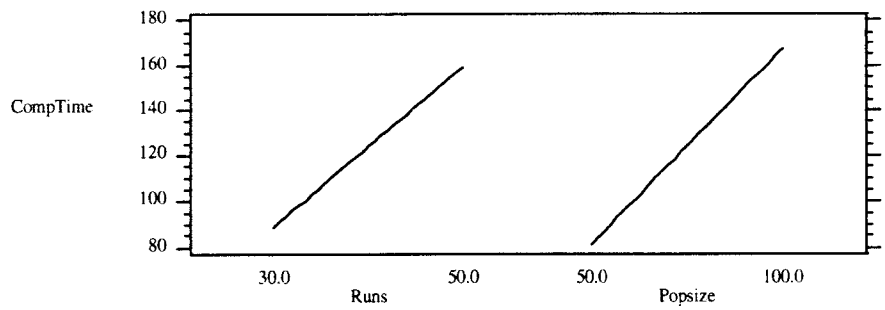


Figure A19 - Main Effects Plot for CompTime

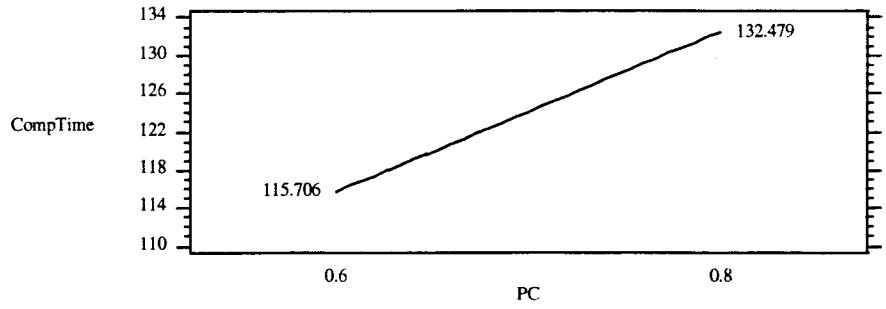


Figure A20 - Interaction Plot for CompTime

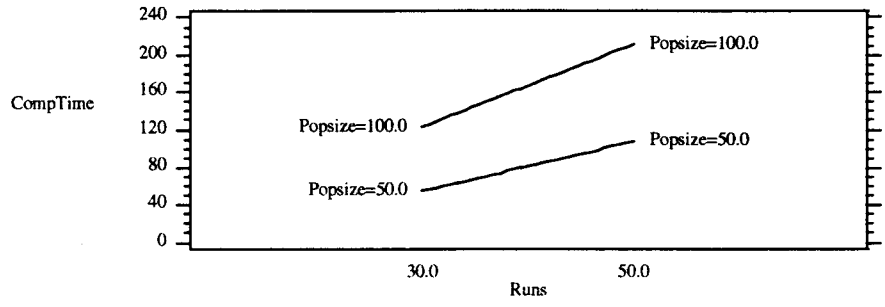


Figure A21 - Interaction Plot for CompTime

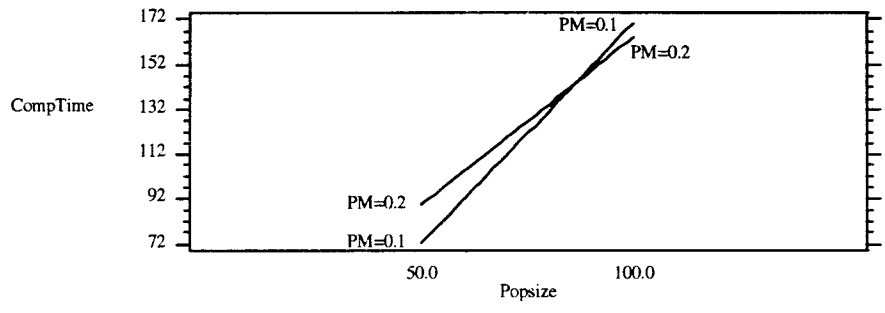


Figure A22 - Interaction Plot for CompTime

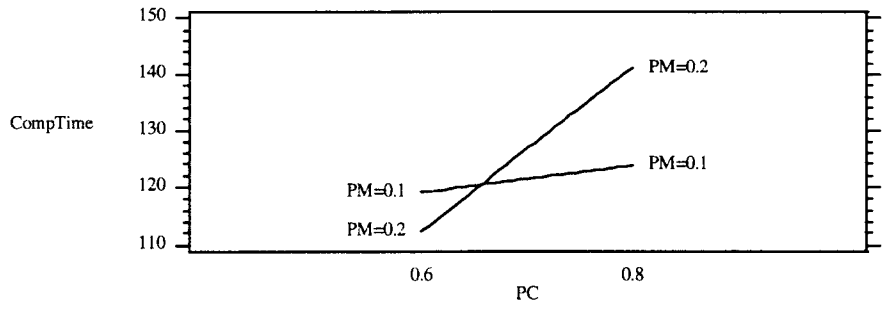


Figure A23 - Normal Probability Plot for Residuals - TotCost Heuristic 3

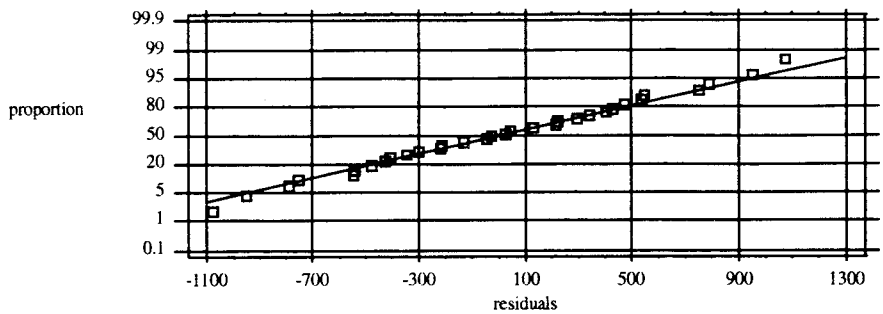


Figure A24 - Residual Plot for TotCost Heuristic 3

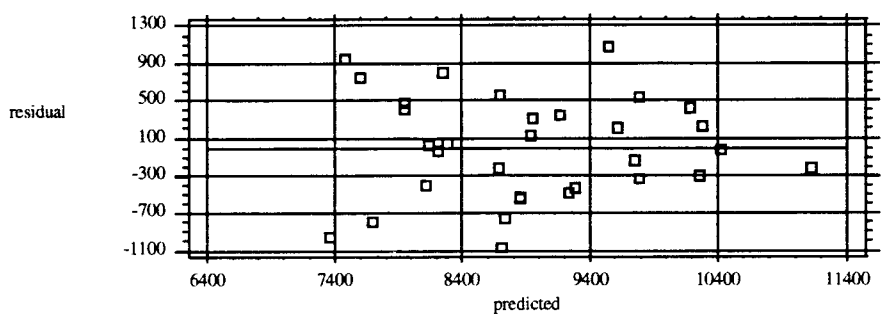


Figure A25 - Main Effects Plot for TotCost

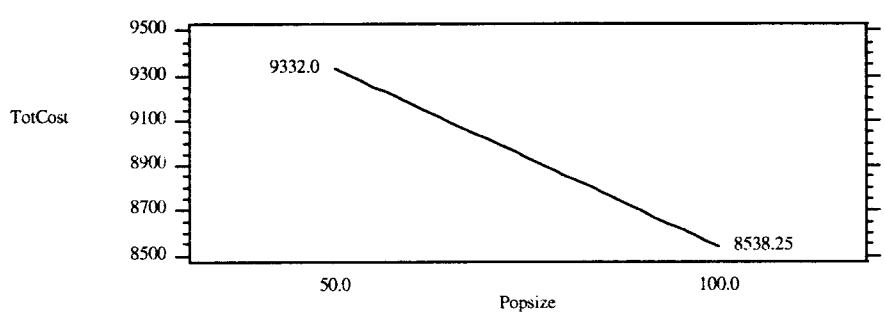


Figure A26 - Interaction Plot for TotCost

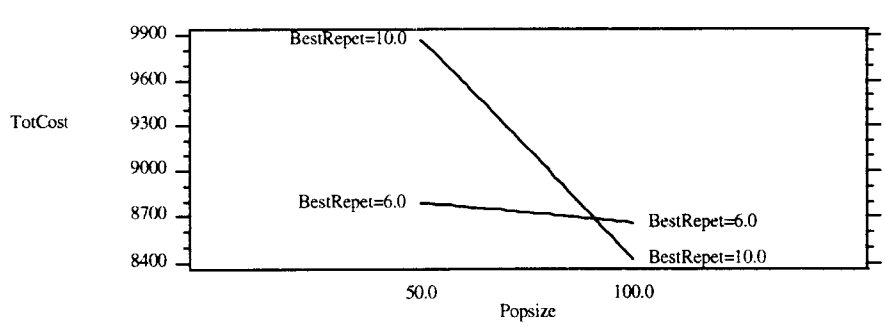


Figure A27 - Interaction Plot for TotCost

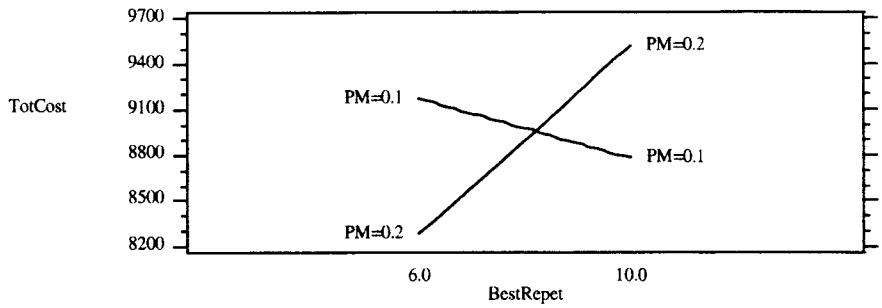


Figure A28 - Normal Probability Plot for Residuals - CompTime Heuristic 3

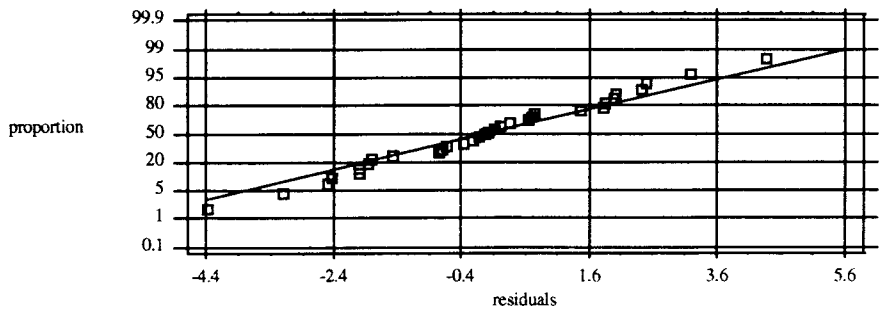


Figure A29 - Residual Plot for CompTime Heuristic 3

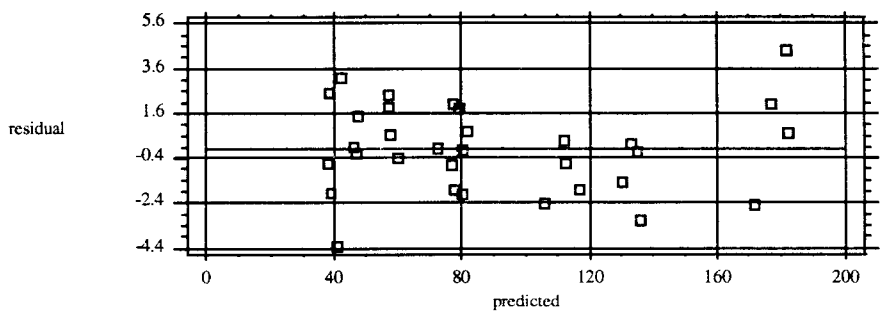


Figure A30 - Main Effects Plot for CompTime

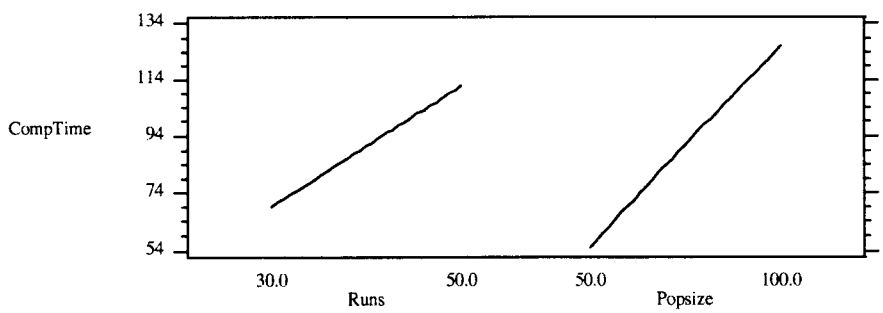


Figure A31 - Main Effects Plot for CompTime

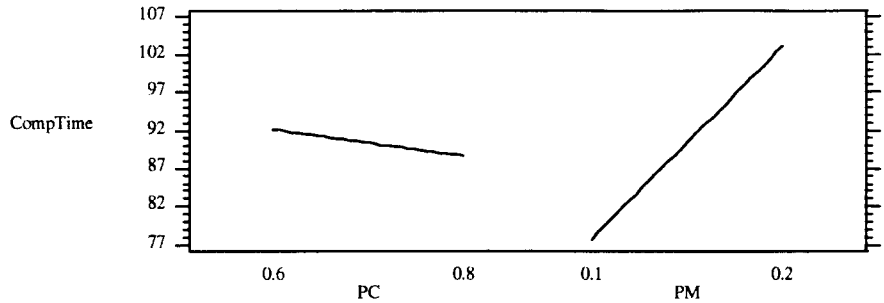


Figure A32 - Interaction Plot for CompTime

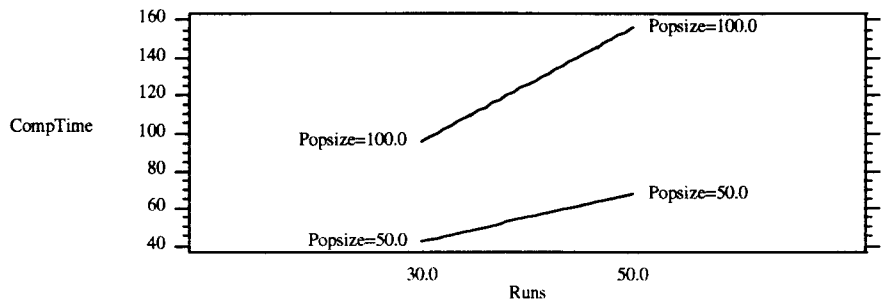


Figure A33 - Interaction Plot for CompTime

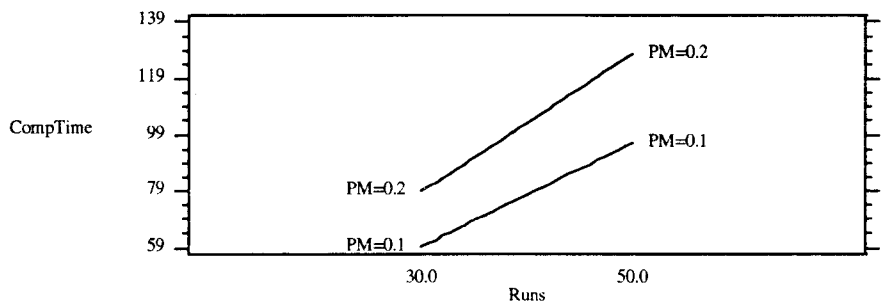
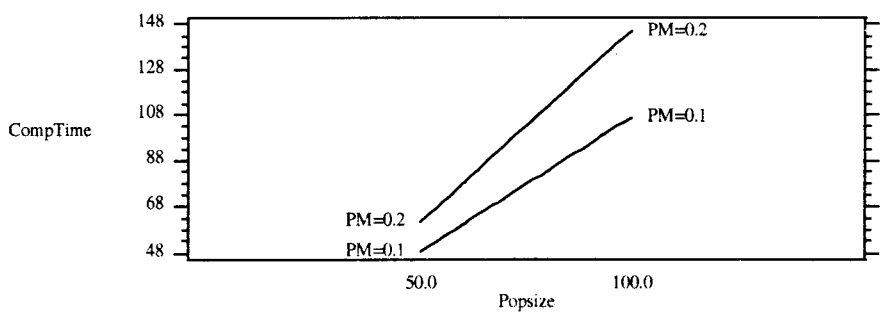


Figure A34 - Interaction Plot for CompTime



APPENDIX B

Experim. Run	Factors				Results		
	GA Runs	Popsize	MutElem	PM	TotCost ($)	CompTime (sec.)	Configuration
1	60	50	6	0.2	8164	125.18	2,2,2,2,2,1,1,0,1,2,2,2,0,2,1,2,2,1,2,1
2	40	100	10	0.2	7139	176.75	2,2,1,2,2,0,1,1,2,0,0,2,0,2,2,2,0,1,2,2
3	40	50	10	0.1	8319	80.36	2,2,2,2,2,2,2,2,0,0,2,2,2,0,0,2,2,1,2,2
4	50	75	8	0.2	6569	157.86	2,2,2,2,2,2,2,2,1,2,0,0,0,2,2,0,2,2,2,2
5	50	75	8	0.2	7323	158.57	2,2,1,2,2,0,1,1,2,0,1,2,2,0,1,2,2,1,2,2
6	50	75	8	0.2	10255	156.92	2,0,2,2,2,0,2,2,1,0,2,1,2,2,1,2,0,2,2,0
7	50	75	8	0.1	7623	147.91	1,2,2,2,2,1,2,2,1,2,1,2,2,2,2,2,2,2,2,2
8	40	50	6	0.1	8669	80.46	2,1,2,2,2,2,1,1,2,0,2,2,2,2,1,0,0,2,2,2
9	70	75	8	0.2	7069	221.52	2,2,2,2,2,0,2,1,2,2,2,0,0,2,2,2,0,2,2,2
10	40	100	6	0.2	7661	173.8	2,2,0,2,2,0,2,2,2,0,2,2,2,0,0,0,2,2,0,2
11	50	75	12	0.2	7406	157.41	2,2,0,2,1,2,1,2,1,2,1,2,2,0,2,2,2,1,2,2
12	50	75	4	0.2	7816	158.63	2,2,1,0,2,0,1,1,2,2,2,2,2,0,1,2,2,2,0,2
13	50	75	8	0.2	8036	159.01	2,2,2,2,2,2,1,1,2,1,2,2,2,2,1,2,2,2,2,1
14	50	75	8	0.2	7975	157.47	2,2,1,2,2,0,1,2,1,0,2,2,0,1,0,2,2,1,2,2
15	50	25	8	0.2	8744	50.47	2,2,0,2,2,0,2,2,2,1,2,0,0,1,1,2,2,2,2,1
16	40	100	10	0.1	6108	168.08	2,2,0,2,2,0,2,2,2,2,2,2,2,2,2,0,2,2,2
17	50	125	8	0.2	7756	268.26	1,2,2,2,2,2,2,0,2,2,1,1,2,2,2,2,2,2,2,2
18	60	50	10	0.2	7209	129.63	2,2,2,2,2,0,2,2,1,0,0,0,2,2,1,0,2,2,2,2
19	60	50	6	0.1	7356	118.7	2,2,1,2,2,0,2,2,1,2,2,2,2,2,0,0,0,1,0,2
20	50	75	8	0.3	7027	166.09	2,1,1,2,2,0,2,2,1,2,2,2,2,2,2,2,0,2,2,2
21	60	100	6	0.2	7370	259.91	2,2,2,2,2,2,2,2,1,2,1,1,2,2,2,2,0,2,2,1
22	50	75	8	0.2	8186	159.29	2,2,1,2,1,2,1,2,2,2,0,0,2,0,1,2,2,2,2,2
23	50	75	8	0.2	7525	157.37	2,2,1,2,2,2,2,0,2,2,1,2,0,0,2,2,0,1,0,1
24	30	75	8	0.2	6339	98.1	2,2,0,2,2,2,2,2,1,2,1,2,2,2,2,0,2,1,0,2
25	40	50	6	0.2	9088	84.96	2,2,0,0,2,2,0,1,2,1,1,2,2,0,2,0,2,1,2,2
26	60	100	10	0.2	6954	262.98	2,2,1,2,2,0,2,2,2,2,1,2,2,2,2,2,0,1,2,1
27	60	100	10	0.1	7743	245.57	2,2,0,0,2,0,2,0,2,2,0,0,0,2,2,2,2,2,2,2
28	40	100	6	0.1	7023	168.35	2,2,1,2,2,0,2,2,1,0,1,2,2,2,2,2,2,1,2,1
29	60	100	6	0.1	5935	245.9	2,2,1,2,2,2,2,2,1,2,1,2,0,2,2,2,2,1,2,2
30	50	75	8	0.2	5898	158.68	2,2,2,2,2,0,2,2,2,2,1,2,0,2,2,2,2,2,2,2
31	50	75	8	0.2	7624	158.96	2,2,2,2,2,2,1,2,2,1,0,0,0,0,2,0,2,2,2,2
32	40	50	10	0.2	7565	87.83	2,2,1,2,2,0,0,2,2,2,0,0,2,2,2,2,0,2,2,2
33	60	50	10	0.1	7544	118.81	2,2,0,2,2,2,2,1,2,2,2,2,2,2,2,0,0,2,0,1
34	50	75	8	0.2	7218	158.68	2,2,0,2,1,2,2,2,1,2,0,2,0,2,1,2,2,1,2,2
35	50	75	8	0.2	6185	158.39	2,2,1,2,2,0,2,2,2,2,2,2,2,2,1,2,2,2,2,2
36	50	75	8	0.2	6599	158.9	2,2,2,2,2,2,2,2,1,0,2,2,2,0,1,2,2,2,0,2

Table B1 - Experimental runs and results Heuristic 1 - Response Surface Design

Table B2 - Analysis of Variance for TotCost - Heuristic 1 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	140607.0	1	140607.0	0.17	0.6860
B:Popsize	4.13091E+06	1	4.13091E+06	4.94	0.0374
C:MutElem	511876.0	1	511876.0	0.61	0.4428
D:PM	66255.0	1	66255.0	0.08	0.7811
AA	1075920.0	1	1075920.0	1.29	0.2695
AB	739170.0	1	739170.0	0.88	0.3579
AC	967764.0	1	967764.0	1.16	0.2943
AD	2889.1	1	2889.1	0.00	0.9537
BB	1.32045E+06	1	1.32045E+06	1.58	0.2228
BC	420877.0	1	420877.0	0.50	0.4859
BD	296208.0	1	296208.0	0.35	0.5581
CC	60233.4	1	60233.4	0.07	0.7910
CD	1074850.0	1	1074850.0	1.29	0.2697
DD	25293.8	1	25293.8	0.03	0.8636
Total Error	1.75652E+07	21	836437.0		
Total (corr.)	2.83985E+07	35			
R-squared	38.1475%				

With lack-of-fit test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	140607.0	1	140607.0	0.11	0.7513
B:Popsize	4.13091E+06	1	4.13091E+06	3.10	0.1059
C:MutElem	511876.0	1	511876.0	0.38	0.5478
D:PM	66255.0	1	66255.0	0.05	0.8276
AA	1075920.0	1	1075920.0	0.81	0.3879
AB	739170.0	1	739170.0	0.56	0.4718
AC	967764.0	1	967764.0	0.73	0.4121
AD	2889.1	1	2889.1	0.00	0.9637
BB	1.32045E+06	1	1.32045E+06	0.99	0.3407
BC	420877.0	1	420877.0	0.32	0.5852
BD	296208.0	1	296208.0	0.22	0.6464
CC	60233.4	1	60233.4	0.05	0.8354
CD	1074850.0	1	1074850.0	0.81	0.3881
DD	25293.8	1	25293.8	0.02	0.8929
Lack-of-fit	2.92129E+06	10	292129	0.22	0.9882
Pure error	1.46439E+07	11	1.33126E+06		
Total (corr.)	2.83985E+07	35			
R-squared	38.1475%				

Table B3 - Analysis of Variance for TotCost - Heuristic 1 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
B:Popsize	4.13091E+06	1	4.13091E+06	6.25	0.0181
AA	1075920.0	1	1075920.0	1.63	0.2118
AC	967764.0	1	967764.0	1.46	0.2357
BB	1.32045E+06	1	1.32045E+06	2.00	0.1678
CD	1074850.0	1	1074850.0	1.63	0.2120
Total Error	1.98286E+07	30	836437.0		
Total (corr.)	2.83985E+07	35			
R-squared	30.1773%				

Table B4 - Analysis of Variance for TotCost - Heuristic 1 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
B:Popsize	4.13091E+06	1	4.13091E+06	5.79	0.0217
Total Error	1.98286E+07	34	713752.0		
Total (corr.)	2.83985E+07	35			
R-squared	14.5462%				

With Lack-of-fit test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
B:Popsize	4.13091E+06	1	4.13091E+06	5.79	0.0234
Lack-of-fit	1.73880E+06	3	579600	0.8	0.5047
Pure error	2.25288E+07	31	1.33126E+06		
Total (corr.)	2.83985E+07	35			
R-squared	14.5462%				

Table B5 - Analysis of Variance for CompTime - Heuristic 1 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	22382.8	1	22382.8	18276.55	0.0000
B:Popsize	71612.3	1	71612.3	58474.71	0.0000
C:MutElem	4.429	1	4.429	3.62	0.0710
D:PM	514.949	1	514.949	420.48	0.0000
AA	2.29873	1	2.29873	1.88	0.1851
AB	1778.1	1	1778.1	1451.90	0.0000
AC	0.213906	1	0.213906	0.17	0.6802
AD	32.0073	1	32.0073	26.14	0.0000
BB	0.78647	1	0.78647	0.64	0.4319
BC	0.228006	1	0.228006	0.19	0.6705
BD	16.5446	1	16.5446	13.51	0.0014
CC	1.03081	1	1.03081	0.84	0.3693
CD	12.1278	1	12.1278	9.90	0.0049
DD	6.04071	1	6.04071	4.93	0.0375
Total Error	257181	21	257181		
Total (corr.)	96389.5	35			
R-squared	99.9733%				

With Lack-of-fit test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	22382.8	1	22382.8	39201.94	0.0000
B:Popsize	71612.3	1	71612.3	125424.21	0.0000
C:MutElem	4.429	1	4.429	7.76	0.0177
D:PM	514.949	1	514.949	901.90	0.0000
AA	2.29873	1	2.29873	4.03	0.0700
AB	1778.1	1	1778.1	3114.22	0.0000
AC	0.213906	1	0.213906	0.37	0.5529
AD	32.0073	1	32.0073	56.06	0.0000
BB	0.78647	1	0.78647	1.38	0.2653
BC	0.228006	1	0.228006	0.40	0.5403
BD	16.5446	1	16.5446	28.98	0.0002
CC	1.03081	1	1.03081	1.81	0.2061
CD	12.1278	1	12.1278	21.24	0.0008
DD	6.04071	1	6.04071	10.58	0.0077
Lack-of-fit	19	10	1.94375	3.4	0.0282
Pure error	6.28057	11	0.570961		
Total (corr.)	96389.5	35			
R-squared	99.9733%				

Table B6 - Analysis of Variance for CompTime - Heuristic 1 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	22382.8	1	22382.8	20000.84	0.0000
B:Popsize	71612.3	1	71612.3	63991.46	0.0000
C:MutElem	4.429	1	4.429	3.96	0.0577
D:PM	514.949	1	514.949	460.15	0.0000
AA	2.29873	1	2.29873	2.05	0.1642
AB	1778.1	1	1778.1	1588.88	0.0000
AD	32.0073	1	32.0073	28.60	0.0000
BD	16.5446	1	16.5446	14.78	0.0007
CD	12.1278	1	12.1278	10.84	0.0030
DD	6.04071	1	6.04071	5.40	0.0286
Total Error	27.9773	25	1.11909		
Total (corr.)	96389.5	35			
R-squared	99.9710%				

Table B7 - Analysis of Variance for CompTime - Heuristic 1 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	22382.8	1	22382.8	19221.56	0.0000
B:Popsize	71612.3	1	71612.3	61498.18	0.0000
C:MutElem	4.429	1	4.429	3.80	0.0620
D:PM	514.949	1	514.949	442.22	0.0000
AB	1778.1	1	1778.1	1526.97	0.0000
AD	32.0073	1	32.0073	27.49	0.0000
BD	16.5446	1	16.5446	14.21	0.0009
CD	12.1278	1	12.1278	10.41	0.0034
DD	6.04071	1	6.04071	5.19	0.0312
Total Error	27.9773	26	1.11909		
Total (corr.)	96389.5	35			
R-squared	99.9686%				

With Lack-of-fit test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	22382.8	1	22382.8	39201.94	0.0000
B:Popsize	71612.3	1	71612.3	125424.21	0.0000
C:MutElem	4.429	1	4.429	7.76	0.0177
D:PM	514.949	1	514.949	901.90	0.0000
AB	1778.1	1	1778.1	3114.22	0.0000
AD	32.0073	1	32.0073	56.06	0.0000
BD	16.5446	1	16.5446	28.98	0.0002
CD	12.1278	1	12.1278	21.24	0.0008
DD	6.04071	1	6.04071	10.58	0.0077
Lack-of-fit	19	15	1.94375	2.8	0.0453
Pure error	6.28057	11	0.570961		
Total (corr.)	96389.5	35			
R-squared	99.9686%				

Experim. Run	Factors					Results		
	GA Runs	Popsize	MutElem	PC	PM	TotCost ($)	CompTime (sec.)	Configuration
1	50	75	8	0.7	0.2	8806	170.73	1,2,0,0,2,2,2,2,1,2,2,1,0,2,1,0,2,2,2,2
2	50	75	8	0.7	0.2	8639	106.44	2,0,0,2,2,2,2,0,2,2,1,2,2,2,2,2,0,2,0,1
3	40	100	10	0.6	0.2	7951	122.26	2,2,0,2,2,2,2,2,1,2,2,2,0,0,2,2,2,0,2,1
4	50	75	4	0.7	0.2	8731	119.08	2,2,2,0,2,0,2,0,1,2,2,0,0,2,1,0,0,2,2,2
5	40	50	10	0.8	0.2	9381	60.96	2,2,1,1,0,2,1,2,1,2,0,1,0,2,2,2,2,1,2,2,
6	50	75	8	0.7	0.2	7738	109.52	2,2,2,2,2,2,2,0,2,0,0,2,2,2,2,2,2,2,2,0
7	50	75	12	0.7	0.2	8561	109.41	2,2,0,2,1,2,2,2,2,0,1,1,0,2,2,2,2,2,0,1
8	40	50	10	0.6	0.1	8184	53.88	2,2,1,2,2,1,2,1,2,2,1,2,2,1,1,2,2,1,2,2
9	50	75	8	0.7	0.2	6951	110.02	2,2,2,2,2,2,2,1,2,2,0,0,2,2,2,2,2,1,2,2
10	50	75	8	0.7	0.3	6675	116.56	2,2,0,2,2,0,2,2,2,2,0,2,2,2,1,2,0,1,2,2
11	60	100	6	0.8	0.1	7066	168.85	2,2,1,2,2,0,2,2,1,0,2,2,2,2,1,2,0,1,0,2
12	50	75	8	0.7	0.2	8641	110.79	2,2,0,2,2,2,2,2,2,2,0,2,2,2,2,2,2,0,0,0
13	40	50	6	0.8	0.1	9513	55.64	2,2,0,2,2,2,0,2,1,0,0,1,2,0,2,2,0,2,0,0
14	30	75	8	0.7	0.2	8595	68.06	2,2,0,2,2,2,2,1,2,1,0,0,0,2,2,1,2,1,0,2
15	50	75	8	0.7	0.2	6273	106.45	2,2,0,2,2,2,2,2,1,0,2,2,0,0,2,2,2,2,0,2
16	50	75	8	0.5	0.2	7742	106.28	2,1,1,2,2,2,2,0,2,0,2,2,0,2,2,0,2,1,2,2
17	50	25	8	0.7	0.2	7627	33.51	2,1,1,2,2,2,2,2,2,1,1,2,0,0,2,2,2,2,2,2
18	50	75	8	0.7	0.2	9070	108.81	2,2,0,2,1,2,1,2,2,2,0,0,2,2,1,1,2,1,2,2
19	60	100	10	0.6	0.1	7090	170.22	2,2,0,2,2,2,2,0,1,0,1,1,0,2,2,2,2,2,2,2
20	60	50	6	0.8	0.2	7398	88.87	2,2,2,2,2,1,2,2,1,2,0,2,0,1,2,2,0,2,2,2
21	40	100	6	0.6	0.1	8919	112.98	2,2,2,2,2,0,1,2,2,1,2,2,0,2,2,1,2,1,2,0
22	40	100	10	0.8	0.1	6277	114.14	2,2,2,2,2,0,1,2,2,2,0,2,2,2,2,2,0,2,2,2
23	40	50	6	0.6	0.2	8019	58.71	2,2,0,1,2,2,2,2,1,2,1,1,2,2,2,0,2,2,0,2
24	60	50	10	0.6	0.2	6846	87.06	2,2,1,2,2,2,1,2,2,0,0,2,0,2,2,0,2,1,0,2
25	60	100	10	0.8	0.2	7913	185.49	2,2,1,2,2,2,2,1,2,1,0,2,2,2,2,0,2,1,2,1
26	50	125	8	0.7	0.2	6338	187.96	2,2,2,2,2,0,2,0,2,2,2,2,2,2,2,0,2,0,2
27	70	75	8	0.7	0.2	6734	151.71	2,2,2,2,2,2,2,2,1,2,0,0,0,2,2,0,2,1,2,2
28	50	75	8	0.7	0.2	7640	107.93	2,2,2,2,2,0,2,1,2,2,2,2,0,2,2,0,2,0,2,2
29	50	75	8	0.7	0.1	7469	96.5	2,2,2,2,1,2,2,0,1,0,2,2,0,2,2,2,2,1,0,2
30	40	100	6	0.8	0.2	6296	123.8	2,2,2,2,2,2,2,0,2,2,0,2,0,2,2,2,2,1,0,2
31	60	50	10	0.8	0.1	9101	79.2	2,2,1,2,2,2,2,1,1,2,0,1,0,1,2,0,2,2,2,0
32	50	75	8	0.9	0.2	7968	109.35	2,2,2,2,2,0,0,0,2,2,1,2,2,2,2,2,0,1,2,1
33	50	75	8	0.7	0.2	7640	112.55	2,2,2,2,2,0,2,1,2,2,2,2,0,2,2,0,2,0,2,2
34	60	50	6	0.6	0.1	7755	79.53	2,2,2,2,2,0,1,2,1,2,1,1,2,2,1,2,2,2,0,1
35	50	75	8	0.7	0.2	7565	102.2	2,2,1,2,2,0,1,2,1,2,1,2,2,0,1,2,0,2,0,1
36	60	100	6	0.6	0.2	7613	182.68	2,2,1,0,2,2,2,2,2,2,1,2,2,2,0,2,2,1,2,2

Table B8 - Experimental runs and results Heuristic 2 - Response Surface Design

Table B9 - Analysis of Variance for TotCost - Heuristic 2 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	2331270.0	1	2331270.0	4.50	0.0510
B:Popsize	3.8801E+06	1	3.8801E+06	7.49	0.0153
C:MutElem	1290.7	1	1290.7	0.00	0.9609
D:PC	43350	1	43350	0.08	0.7764
E:PM	692241.0	1	692241.0	1.34	0.2659
AA	8192.0	1	8192.0	0.02	0.9016
AB	1121480.0	1	1121480.0	2.16	0.1620
AC	268324.0	1	268324.0	0.52	0.4829
AD	893025.0	1	893025.0	1.72	0.209
AE	1.0	1	1.0	0.00	0.9989
BB	1113030.0	1	1113030.0	2.15	0.1634
BC	138756.0	1	138756.0	0.27	0.6124
BD	4633260.0	1	4633260.0	8.94	0.0092
BE	693056.0	1	693056.0	1.34	0.2656
CC	1683610.0	1	1683610.0	3.25	0.0916
CD	1.34212E+06	1	1.34212E+06	2.59	0.1284
CE	1.79962E+06	1	1.79962E+06	3.47	0.0821
DD	32004.5	1	32004.5	0.06	0.8071
DE	18906.3	1	18906.3	0.04	0.8511
EE	861985	1	861985	1.66	0.2167
Total Error	7.77436E+06	15	518290.0		
Total (corr.)	2.9330E+07	35			
R-squared	73.4935%				

With Lack-of-fit test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	2331270.0	1	2331270.0	2.95	0.1199
B:Popsize	3.8801E+06	1	3.8801E+06	4.91	0.0538
C:MutElem	1290.7	1	1290.7	0.00	0.9686
D:PC	43350	1	43350	0.05	0.82
E:PM	692241.0	1	692241.0	0.88	0.3735
AA	8192.0	1	8192.0	0.01	0.9211
AB	1121480.0	1	1121480.0	1.42	0.2638
AC	268324.0	1	268324.0	0.34	0.5742
AD	893025.0	1	893025.0	1.13	0.3153
AE	1.0	1	1.0	0.00	0.9991
BB	1113030.0	1	1113030.0	1.41	0.2655
BC	138756.0	1	138756.0	0.18	0.6849
BD	4633260.0	1	4633260.0	5.87	0.0385
BE	693056.0	1	693056.0	0.88	0.3733
CC	1683610.0	1	1683610.0	2.13	0.1782
CD	1.34212E+06	1	1.34212E+06	1.7	0.2247
CE	1.79962E+06	1	1.79962E+06	2.28	0.1654
DD	32004.5	1	32004.5	0.04	0.8449
DE	18906.3	1	18906.3	0.02	0.8804
EE	861985	1	861985	1.09	0.3233
Lack-of-fit	6.68156E+05	6	111359	0.14	0.9865
Pure error	7.1062E+06	9	789578.0		
Total (corr.)	2.933E+07	35			
R-squared	73.4935%				

Table B10 - Analysis of Variance for TotCost - Heuristic 2 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
B:Popsize	3.8801E+06	1	3.8801E+06	7.69	0.0088
AD	893025.0	1	893025.0	1.84	0.1865
BB	1113030.0	1	1113030.0	2.29	0.1418
BD	4633260.0	1	4633260.0	9.53	0.0046
CC	1683610.0	1	1683610.0	3.46	0.0736
CD	1.34212E+06	1	1.34212E+06	2.76	0.1081
CE	1.79962E+06	1	1.79962E+06	3.7	0.0649
EE	861985	1	861985	1.77	0.1941
Total Error	1.3115E+07	27	504424.0		
Total (corr.)	2.933E+07	35			
R-squared	55.2566%				

Table B11 - Analysis of Variance for TotCost - Heuristic 2 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
B:Popsize	3.8801E+06	1	3.8801E+06	6.15	0.0184
BD	4633260.0	1	4633260.0	7.34	0.0106
Total Error	2.0817E+07	33	504424.0		
Total (corr.)	2.933E+07	35			
R-squared	29.0261%				

With Lack-of-fit test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
B:Popsize	3.8801E+06	1	3.8801E+06	5.83	0.0223
BD	4633260.0	1	4633260.0	6.96	0.0133
Lack-of-fit	1.50999E+06	8	517037.0	0.57	0.6885
Pure error	1.93066E+07	33	665746.0		
Total (corr.)	2.93300E+07	35			
R-squared	29.0261%				

Table B12 - Analysis of Variance for CompTime - Heuristic 2 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	10703.2	1	10703.2	1272.02	0.0000
B:Popsize	35687.3	1	35687.3	4241.24	0.0000
C:MutElem	123123.0	1	123123.0	1.46	0.2451
D:PC	10.3622	1	10.3622	1.23	0.2846
E:PM	555.94	1	555.94	66.07	0.0000
AA	0.458403	1	0.458403	0.05	0.8186
AB	1033.46	1	1033.46	122.82	0.0000
AC	0.232806	1	0.232806	0.03	0.8701
AD	0.897756	1	0.897756	0.11	0.7485
AE	18.5115	1	18.5115	2.20	0.1587
BB	3.53115	1	3.53115	0.42	0.5269
BC	1.85641	1	1.85641	0.22	0.6453
BD	0.113906	1	0.113906	0.01	0.9089
BE	26.7548	1	26.7548	3.18	0.0948
CC	46.827	1	46.827	5.57	0.0323
CD	0.604506	1	0.604506	0.07	0.7923
CE	0.100806	1	0.100806	0.01	0.9143
DD	5.06415	1	5.06415	0.6	0.4499
DE	3.23101	1	3.23101	0.38	0.5448
EE	16.5456	1	16.5456	1.97	0.1812
Total Error	126.215	15	8.41434		
Total (corr.)	48253.5	35			
R-squared	99.7384%				

With Lack-of-fit test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	10703.2	1	10703.2	1136.94	0.0000
B:Popsize	35687.3	1	35687.3	3790.87	0.0000
C:MutElem	123123.0	1	123123.0	1.31	0.2823
D:PC	10.3622	1	10.3622	1.1	0.3215
E:PM	555.94	1	555.94	59.05	0.0000
AA	0.458403	1	0.458403	0.05	0.8303
AB	1033.46	1	1033.46	109.78	0.0000
AC	0.232806	1	0.232806	0.02	0.8785
AD	0.897756	1	0.897756	0.10	0.7645
AE	18.5115	1	18.5115	1.97	0.1944
BB	3.53115	1	3.53115	0.38	0.5554
BC	1.85641	1	1.85641	0.20	0.6675
BD	0.113906	1	0.113906	0.01	0.9148
BE	26.7548	1	26.7548	2.84	0.1261
CC	46.827	1	46.827	4.97	0.0527
CD	0.604506	1	0.604506	0.06	0.8056
CE	0.100806	1	0.100806	0.01	0.9199
DD	5.06415	1	5.06415	0.54	0.4820
DE	3.23101	1	3.23101	0.34	0.5724
EE	16.5456	1	16.5456	1.76	0.2176
Lack-of-fit	41.4891	6	6.91484	0.73	0.6349
Pure error	84.7261	9	9.4		
Total (corr.)	48253.5	35			
R-squared	99.7384%				

Table B13 - Analysis of Variance for CompTime - Heuristic 2 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	10703.2	1	10703.2	1751.64	0.0000
B:Popsize	35687.3	1	35687.3	5840.42	0.0000
E:PM	555.94	1	555.94	90.98	0.0000
AB	1033.46	1	1033.46	169.13	0.0000
AE	18.5115	1	18.5115	3.03	0.0931
BE	26.7548	1	26.7548	4.38	0.0459
CC	46.827	1	46.827	7.66	0.0101
EE	16.5456	1	16.5456	2.71	0.1115
Total Error	164.981	27	6.11039		
Total (corr.)	48253.5	35			
R-squared	99.7384%				

Table B14 - Analysis of Variance for CompTime - Heuristic 2 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	10703.2	1	10703.2	1551.67	0.0000
B:Popsize	35687.3	1	35687.3	5173.68	0.0000
E:PM	555.94	1	555.94	80.60	0.0000
AB	1033.46	1	1033.46	149.82	0.0000
BE	26.7548	1	26.7548	3.88	0.0585
CC	46.827	1	46.827	6.79	0.0143
Total Error	200.038	29	6.89785		
Total (corr.)	48253.5	35			
R-squared	99.5854%				

Table B15 - Analysis of Variance for CompTime - Heuristic 2 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	10703.2	1	10703.2	1415.81	0.0000
B:Popsize	35687.3	1	35687.3	4720.70	0.0000
E:PM	555.94	1	555.94	73.54	0.0000
AB	1033.46	1	1033.46	136.71	0.0000
CC	46.827	1	46.827	6.19	0.0186
Total Error	226.792	30	7.55975		
Total (corr.)	48253.5	35			
R-squared	99.53%				

With Lack-of-fit test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	10703.2	1	10703.2	1415.81	0.0000
B:Popsize	35687.3	1	35687.3	4720.70	0.0000
E:PM	555.94	1	555.94	73.54	0.0000
AB	1033.46	1	1033.46	136.71	0.0000
CC	46.827	1	46.827	6.19	0.0186
Lack-of-fit	76.8946	10	7.68946	1.03	0.4572
Pure error	149.898	20	7.49489		
Total (corr.)	48253.5	35			
R-squared	99.53%				

Experim. Run	Factors					Results		
	GA Runs	Popsize	BestRepet	PC	PM	TotCost ($)	CompTime (sec.)	Configuration
1	50	75	8	0.7	0.2	9057	104.31	1,2,1,2,1,0,2,1,1,2,2,1,0,0,2,2,2,2,0,2
2	60	100	6	0.8	0.1	8864	152.14	2,2,0,0,2,0,2,1,2,2,2,2,2,0,2,2,0,0,0,2
3	30	75	8	0.7	0.2	8493	63.94	2,2,2,2,2,0,1,0,1,1,2,1,0,2,0,0,2,2,2,2
4	50	75	8	0.5	0.2	8322	111.66	2,0,2,2,2,2,2,2,2,2,2,1,0,0,2,0,2,2,0,2
5	50	75	8	0.7	0.2	7538	103.81	2,2,1,2,1,2,2,2,2,0,1,2,0,2,1,2,2,2,0,2
6	60	50	10	0.8	0.1	10082	69.15	2,2,0,1,0,0,2,2,1,0,0,1,2,2,1,2,2,1,0,2
7	50	75	8	0.7	0.2	6910	105.79	2,2,1,2,2,0,0,2,2,2,1,2,0,0,2,2,2,1,0,2
8	60	100	6	0.6	0.2	8908	222.29	2,2,2,2,2,0,0,0,2,2,1,0,1,0,2,2,2,2,2,2
9	40	100	10	0.8	0.1	7940	102.21	2,1,1,2,2,0,1,2,1,0,1,2,0,0,2,2,0,2,2,2
10	40	50	10	0.6	0.1	10111	48.88	2,2,2,1,1,0,1,0,2,2,0,1,2,0,2,2,2,2,2,1
11	50	75	8	0.7	0.2	9663	105.95	2,2,0,1,0,2,1,0,1,2,0,2,0,2,1,2,0,1,2,2
12	60	100	10	0.6	0.1	7318	157.8	2,2,0,2,2,2,0,2,1,2,0,1,2,2,2,2,2,1,2,2
13	50	75	8	0.7	0.2	6665	108.48	2,2,2,2,2,2,2,2,2,0,2,2,2,2,2,0,0,2,0,2
14	40	50	6	0.8	0.1	11795	48.88	1,2,1,2,0,2,2,1,2,2,2,0,0,0,0,1,2,2,0,1
15	40	100	10	0.6	0.2	7812	147.86	2,2,0,0,2,2,1,1,2,2,1,2,2,2,1,2,2,2,0,2
16	50	75	8	0.7	0.2	7908	106.94	2,2,1,2,0,0,1,0,2,2,2,2,2,2,2,2,0,2,2,2
17	50	75	12	0.7	0.2	9057	104.36	1,2,1,2,1,0,2,1,1,2,2,1,0,0,2,2,2,2,0,2
18	40	100	6	0.8	0.2	8902	132.38	2,1,1,2,2,0,0,1,2,2,0,0,0,2,2,2,2,1,0,2
19	40	50	10	0.8	0.2	10741	59.7	1,2,2,2,1,0,0,2,1,0,2,1,0,0,1,1,2,2,2,2
20	40	50	6	0.6	0.2	9699	60.69	2,2,2,0,2,1,0,2,1,0,2,1,2,0,1,2,0,2,2,2
21	70	75	8	0.7	0.2	8447	143.74	2,1,0,2,2,2,2,1,1,2,0,1,2,2,2,2,0,1,0,2
22	60	100	10	0.8	0.2	8535	202.18	2,1,1,2,2,0,2,1,2,2,0,2,0,2,2,2,2,2,2,0
23	50	75	8	0.9	0.2	9151	94.63	2,1,0,2,2,0,0,0,1,2,2,2,2,2,0,0,2,1,0,2
24	50	25	8	0.7	0.2	11324	29.93	2,0,0,2,1,0,1,2,2,2,2,0,2,2,2,2,0,1,2,0
25	50	75	8	0.7	0.3	8405	149.89	2,2,2,2,2,2,1,2,2,0,0,2,2,2,1,0,0,0,2,2
26	50	75	8	0.7	0.2	8372	106.17	2,2,1,0,2,2,2,1,1,2,1,0,2,2,1,2,2,1,0,2
27	50	75	4	0.7	0.2	10008	104.91	2,2,2,0,2,0,0,1,1,2,2,2,0,2,0,1,2,2,2,1
28	60	50	10	0.6	0.2	8965	89.64	2,2,2,2,2,2,1,2,0,2,2,0,0,0,2,1,2,1,0,2
29	60	50	6	0.8	0.2	9662	85.85	2,2,1,2,2,1,1,2,2,0,0,2,2,1,1,0,1,1,0,2
30	50	75	8	0.7	0.1	9198	87.49	2,2,0,2,0,0,2,0,2,0,1,1,0,2,1,0,2,2,0,2
31	50	125	8	0.7	0.2	6748	196.8	2,2,2,2,2,0,2,1,1,2,2,2,0,0,2,0,2,1,0,2
32	50	75	8	0.7	0.2	8775	105.68	2,2,1,2,2,0,2,2,2,2,1,2,1,0,0,2,2,1,0,1
33	60	50	6	0.6	0.1	9778	68.77	2,2,2,1,2,2,2,0,1,2,2,2,0,0,2,1,2,1,2,0
34	50	75	8	0.7	0.2	9292	104.85	2,2,2,1,2,0,2,2,1,2,1,0,0,2,2,2,2,1,2,0
35	40	100	6	0.6	0.1	7415	105.79	2,2,0,2,2,0,1,1,1,2,1,0,0,2,2,2,2,2,0,2
36	50	75	8	0.7	0.2	7632	106.28	2,2,2,0,2,2,1,1,2,0,2,2,2,0,2,2,2,1,2,2

Table B16 - Experimental runs and results Heuristic 3 - Response Surface Design

Table B17- Analysis of Variance for TotCost - Heuristic 3 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	239001.0	1	239001.0	0.36	0.5587
B:Popsize	2.4586E+07	1	2.4586E+07	36.80	0.0000
C:BestRepet	1224470.0	1	1224470.0	1.83	0.1959
D:PC	2.78E+06	1	2.78E+06	4.17	0.0593
E:PM	115509.0	1	115509.0	0.17	0.6835
AA	215441.0	1	215441.0	0.32	0.5785
AB	1832640.0	1	1832640.0	2.74	0.1185
AC	76314.1	1	76314.1	0.11	0.7401
AD	293493.0	1	293493.0	0.44	0.5175
AE	1139.1	1	1139.1	0.00	0.9676
BB	1599220.0	1	1599220.0	2.39	0.1427
BC	131225.0	1	131225.0	0.20	0.6640
BD	55107.6	1	55107.6	0.08	0.7779
BE	1768240.0	1	1768240.0	2.65	0.1246
CC	3861400.0	1	3868140.0	5.79	0.0295
CD	6.84756E+03	1	6.84756E+03	0.01	0.9207
CE	1.02881E+05	1	1.02881E+05	0.15	0.7003
DD	707356	1	707356	1.06	0.3198
DE	160601	1	160601	0.24	0.6310
EE	870430	1	870430	1.3	0.2716
Total Error	1.00224E+07	15	668159.0		
Total (corr.)	5.0659E+07	35			
R-squared	80.2161%				

With Lack-of-fit test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	239001.0	1	239001.0	0.23	0.6419
B:Popsize	2.4586E+07	1	2.4586E+07	23.82	0.0009
C:BestRepet	1224470.0	1	1224470.0	1.19	0.3044
D:PC	2.78E+06	1	2.78E+06	2.7	0.1350
E:PM	115509.0	1	115509.0	0.11	0.7457
AA	215441.0	1	215441.0	0.21	0.6586
AB	1832640.0	1	1832640.0	1.78	0.2155
AC	76314.1	1	76314.1	0.07	0.7918
AD	293493.0	1	293493.0	0.28	0.6068
AE	1139.1	1	1139.1	0.00	0.9742
BB	1599220.0	1	1599220.0	1.55	0.2447
BC	131225.0	1	131225.0	0.13	0.7296
BD	55107.6	1	55107.6	0.05	0.8224
BE	1768240.0	1	1768240.0	1.71	0.2230
CC	3861400.0	1	3868140.0	3.75	0.0849
CD	6.84756E+03	1	6.84756E+03	0.01	0.9369
CE	1.02881E+05	1	1.02881E+05	0.1	0.7594
DD	707356	1	707356	0.69	0.4292
DE	160601	1	160601	0.16	0.7024
EE	870430	1	870430	0.84	0.3824
Lack-of-fit	7.31965E+05	6	121994	0.12	0.9914
Pure error	9.2904E+06	9	1032270.0		
Total (corr.)	5.066E+07	35			
R-squared	80.2161%				

Table B18- Analysis of Variance for TotCost - Heuristic 3 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
B:Popsize	2.4586E+07	1	2.4586E+07	50.13	0.0000
D:PC	2.78E+06	1	2.78E+06	5.68	0.0240
AB	1832640.0	1	1832640.0	3.74	0.0630
BB	1599220.0	1	1599220.0	3.26	0.0813
BE	1768240.0	1	1768240.0	3.61	0.0676
CC	3861400.0	1	3868140.0	7.89	0.0088
Total Error	1.42222E+07	29	490420.0		
Total (corr.)	5.0659E+07	35			
R-squared	71.9257%				

Table B19- Analysis of Variance for TotCost - Heuristic 3 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
B:Popsize	2.4586E+07	1	2.4586E+07	40.51	0.0000
D:PC	2.78E+06	1	2.78E+06	4.59	0.0400
CC	3861400.0	1	3868140.0	6.37	0.0167
Total Error	1.94223E+07	32	606946.0		
Total (corr.)	5.0659E+07	35			
R-squared	61.6609%				

With Lack-of-fit test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
B:Popsize	2.4586E+07	1	2.4586E+07	39.37	0.0000
D:PC	2.78E+06	1	2.78E+06	4.46	0.0445
CC	3861400.0	1	3868140.0	6.19	0.0195
Lack-of-fit	3.18475E+06	6	530792	0.85	0.5437
Pure error	1.6238E+07	26	624520.0		
Total (corr.)	5.066E+07	35			
R-squared	61.6609%				

Table B20- Analysis of Variance for CompTime - Heuristic 3 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	10459.6	1	10459.6	3429.15	0.0000
B:Popsize	4.3762E+04	1	4.3762E+04	14347.06	0.0000
C:BestRepet	0.00920417	1	0.00920417	0.00	0.9569
D:PC	2.89E+02	1	2.89E+02	94.76	0.0000
E:PM	5758.9	1	5758.9	1888.03	0.0000
AA	6.98445	1	6.98445	2.29	0.1510
AB	1423.36	1	1423.36	466.65	0.0000
AC	28.0635	1	28.0635	9.20	0.0084
AD	5.20981	1	5.20981	1.71	0.2109
AE	204.705	1	204.705	67.11	0.0000
BB	117.236	1	117.236	38.44	0.0000
BC	2.05206	1	2.05206	0.67	0.4249
BD	102.162	1	102.162	33.49	0.0000
BE	1001.25	1	1001.25	328.26	0.0000
CC	2.30588	1	2.30588	0.76	0.3983
CD	46.75140	1	46.75140	15.33	0.0014
CE	1.15026	1	1.15026	0.38	0.5484
DD	13.1456	1	13.1456	4.31	0.0555
DE	62.055	1	62.055	20.34	0.0004
EE	337.026	1	337.026	110.49	0.0000
Total Error	45.75310	15	3.05021		
Total (corr.)	63668.3	35			
R-squared	99.9281%				

With Lack-of-fit test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	10459.6	1	10459.6	5897.46	0.0000
B:Popsize	4.3762E+04	1	4.3762E+04	24674.09	0.0000
C:BestRepet	0.00920417	1	0.00920417	0.01	0.9441
D:PC	2.89E+02	1	2.89E+02	162.98	0.0000
E:PM	5758.9	1	5758.9	3247.03	0.0000
AA	6.98445	1	6.98445	3.94	0.0785
AB	1423.36	1	1423.36	802.54	0.0000
AC	28.0635	1	28.0635	15.82	0.0032
AD	5.20981	1	5.20981	2.94	0.1207
AE	204.705	1	204.705	115.42	0.0000
BB	117.236	1	117.236	66.10	0.0000
BC	2.05206	1	2.05206	1.16	0.3101
BD	102.162	1	102.162	57.60	0.0000
BE	1001.25	1	1001.25	564.53	0.0000
CC	2.30588	1	2.30588	1.30	0.2836
CD	46.75140	1	46.75140	26.36	0.0006
CE	1.15026	1	1.15026	0.65	0.4414
DD	13.1456	1	13.1456	7.41	0.0235
DE	62.055	1	62.055	34.99	0.0002
EE	337.026	1	337.026	190.03	0.0000
Lack-of-fit	2.97909E+01	6	4.96514	2.8	0.0803
Pure error	1.5962E+01	9	1.8		
Total (corr.)	63668.3	35			
R-squared	99.9281%				

Table B21- Analysis of Variance for CompTime - Heuristic 3 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	10459.6	1	10459.6	3703.81	0.0000
B:Popsize	4.3762E+04	1	4.3762E+04	15496.20	0.0000
D:PC	2.89E+02	1	2.89E+02	102.35	0.0000
E:PM	5758.9	1	5758.9	2039.25	0.0000
AA	6.98445	1	6.98445	2.47	0.1315
AB	1423.36	1	1423.36	504.02	0.0000
AC	28.0635	1	28.0635	9.94	0.0050
AE	204.705	1	204.705	72.49	0.0000
BB	117.236	1	117.236	41.51	0.0000
BD	102.162	1	102.162	36.18	0.0000
BE	1001.25	1	1001.25	354.55	0.0000
CD	46.75140	1	46.75140	16.55	0.0006
DD	13.1456	1	13.1456	4.65	0.0433
DE	62.055	1	62.055	21.97	0.0001
EE	337.026	1	337.026	119.34	0.0000
Total Error	56.48030	20	2.82402		
Total (corr.)	63668.3	35			
R-squared	99.9113%				

Table B22- Analysis of Variance for CompTime - Heuristic 3 - Response Surface Design

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	10459.6	1	10459.6	3461.01	0.0000
B:Popsize	4.3762E+04	1	4.3762E+04	14480.35	0.0000
D:PC	2.89E+02	1	2.89E+02	95.64	0.0000
E:PM	5758.9	1	5758.9	1905.57	0.0000
AB	1423.36	1	1423.36	470.98	0.0000
AC	28.0635	1	28.0635	9.29	0.0061
AE	204.705	1	204.705	67.74	0.0000
BB	117.236	1	117.236	38.79	0.0000
BD	102.162	1	102.162	33.80	0.0000
BE	1001.25	1	1001.25	331.31	0.0000
CD	46.75140	1	46.75140	15.47	0.0008
DD	13.1456	1	13.1456	4.35	0.0494
DE	62.055	1	62.055	20.53	0.0002
EE	337.026	1	337.026	111.52	0.0000
Total Error	63.46480	21	3.02213		
Total (corr.)	63668.3	35			
R-squared	99.9003%				

Table B22 (continued) - With Lack-of-fit test

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:Runs	10459.6	1	10459.6	5897.46	0.0000
B:Popsize	4.3762E+04	1	4.3762E+04	24674.09	0.0000
D:PC	2.89E+02	1	2.89E+02	162.98	0.0000
E:PM	5758.9	1	5758.9	3247.03	0.0000
AB	1423.36	1	1423.36	802.54	0.0000
AC	28.0635	1	28.0635	15.82	0.0032
AE	204.705	1	204.705	115.42	0.0000
BB	117.236	1	117.236	66.10	0.0000
BD	102.162	1	102.162	57.60	0.0000
BE	1001.25	1	1001.25	564.53	0.0000
CD	46.75140	1	46.75140	26.36	0.0006
DD	13.1456	1	13.1456	7.41	0.0235
DE	62.055	1	62.055	34.99	0.0002
EE	337.026	1	337.026	190.03	0.0000
Lack-of-fit	4.75025E+01	12	3.95854	2.23	0.1172
Pure error	1.5962E+01	9	1.8		
Total (corr.)	6.367E+04	35			
R-squared	99.9003%				

Figure B1 - Normal Probability Plot for Residuals - TotCost Heuristic 1

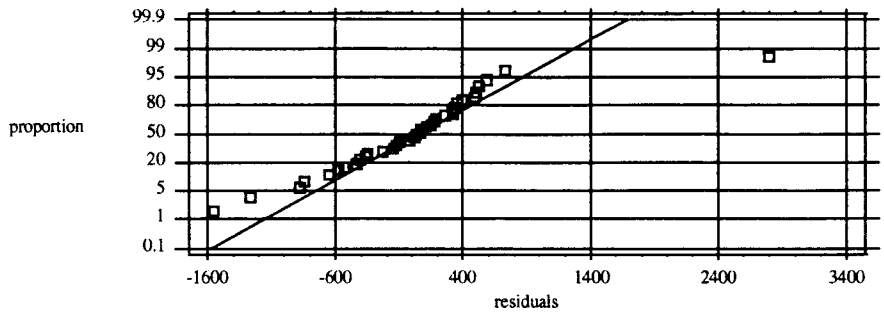


Figure B2 - Residual Plot for TotCost Heuristic 1

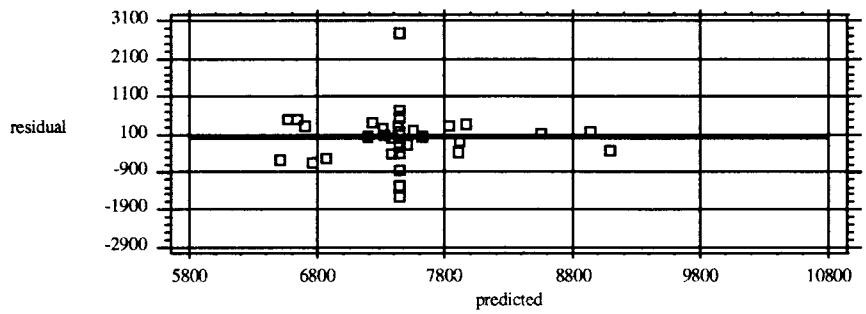


Figure B3 - Main Effects Plot for TotCost

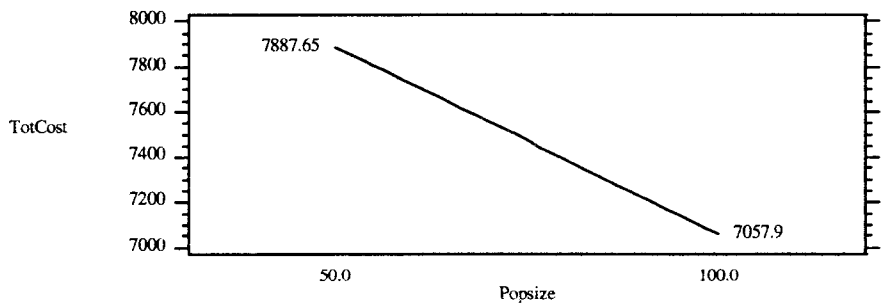


Figure B4 - Normal Probability Plot for Residuals - CompTime Heuristic 1

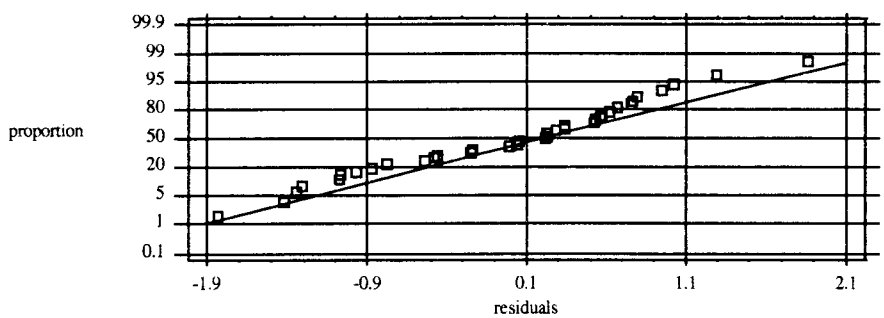


Figure B5 - Residual Plot for CompTime Heuristic 1

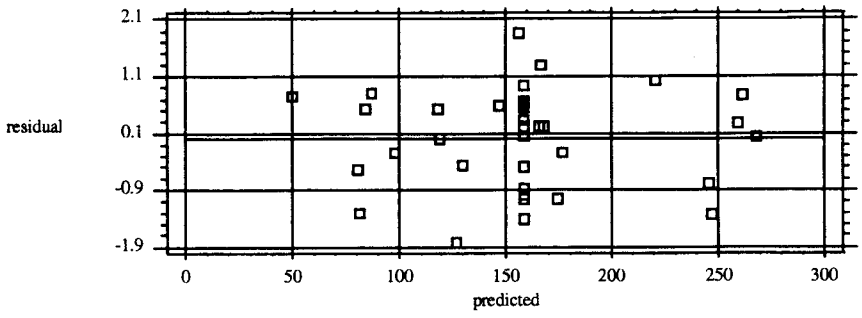


Figure B6 - Main Effects Plot for CompTime

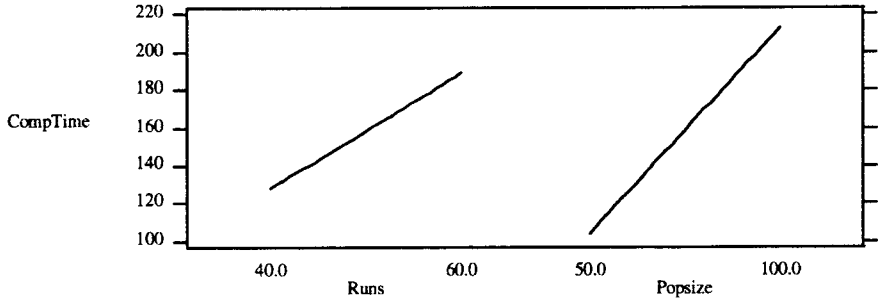


Figure B7 - Main Effects Plot for CompTime

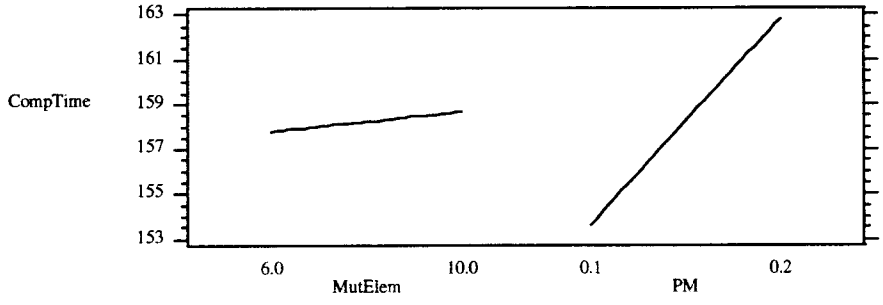


Figure B8 - Interaction Plot for CompTime

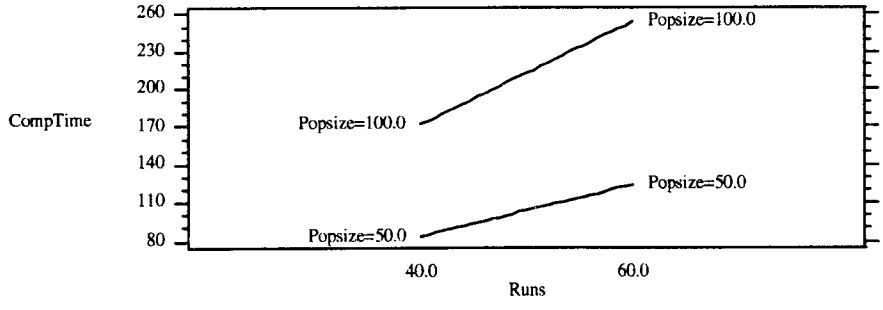


Figure B9 - Interaction Plot for CompTime

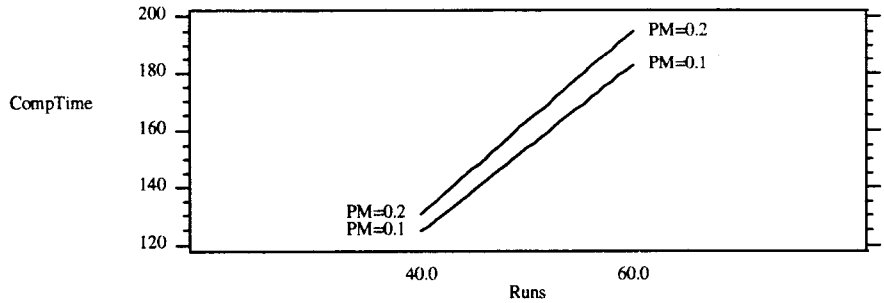


Figure B10 - Interaction Plot for CompTime

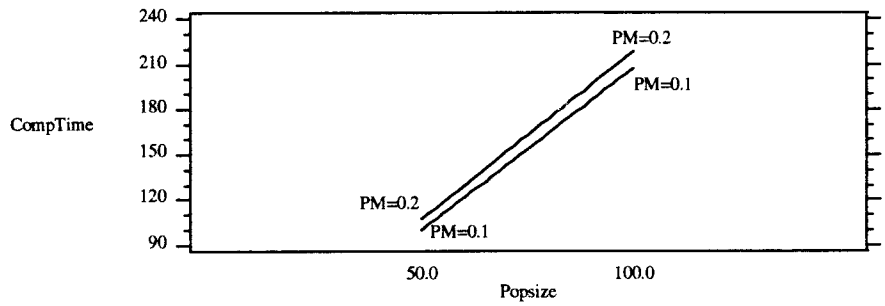


Figure B11 - Interaction Plot for CompTime

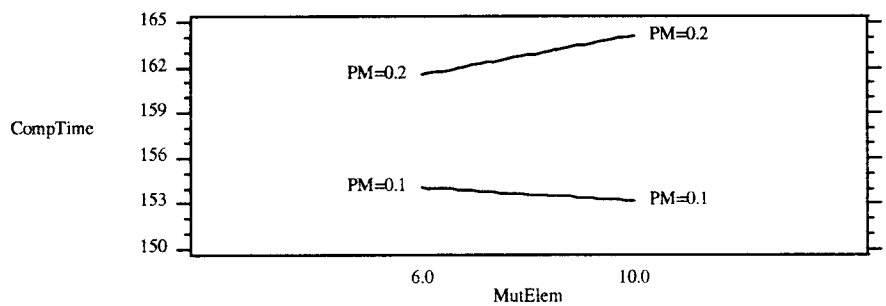


Figure B12 - Estimated Response Surface

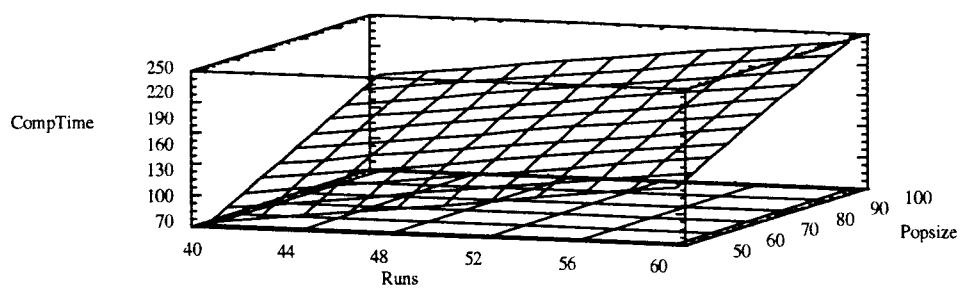


Figure B13 - Estimated Response Surface

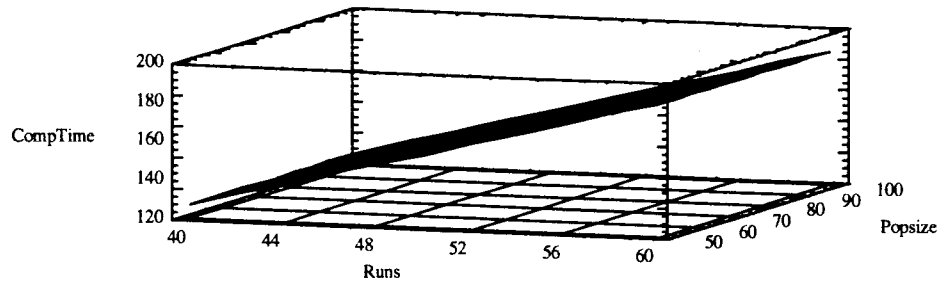


Figure B14 - Estimated Response Surface

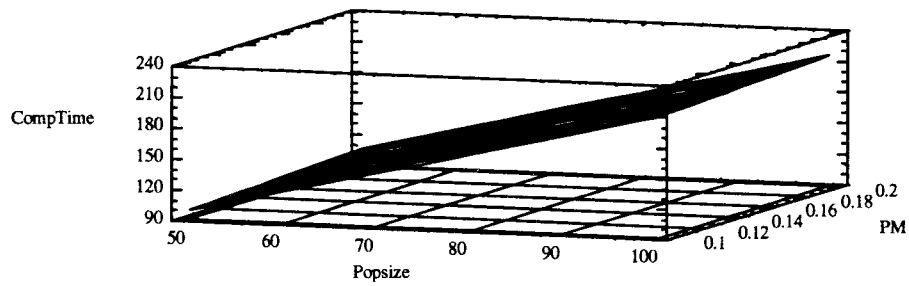


Figure B15 - Estimated Response Surface

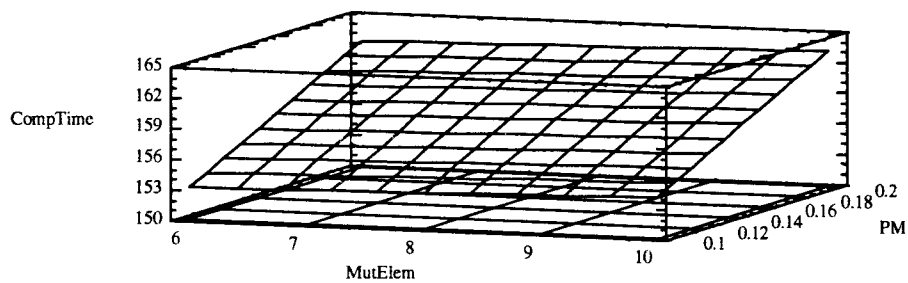


Figure B16 - Normal Probability Plot for Residuals - TotCost Heuristic 2

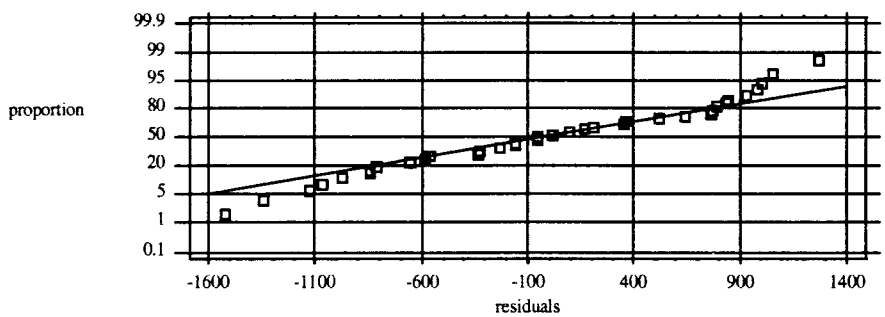


Figure B17 - Residual Plot for TotCost Heuristic 2

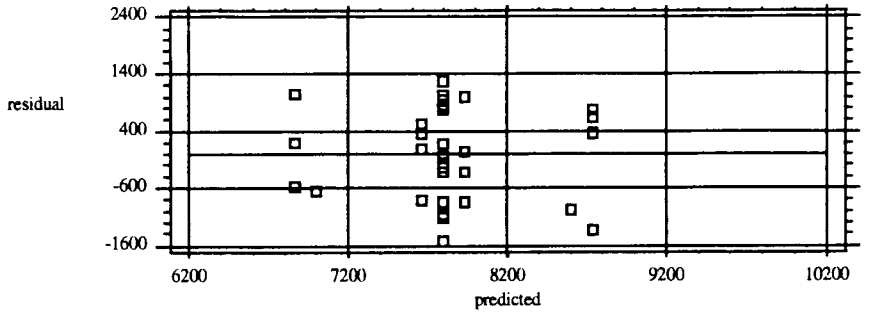


Figure B18 - Interaction Plot for TotCost

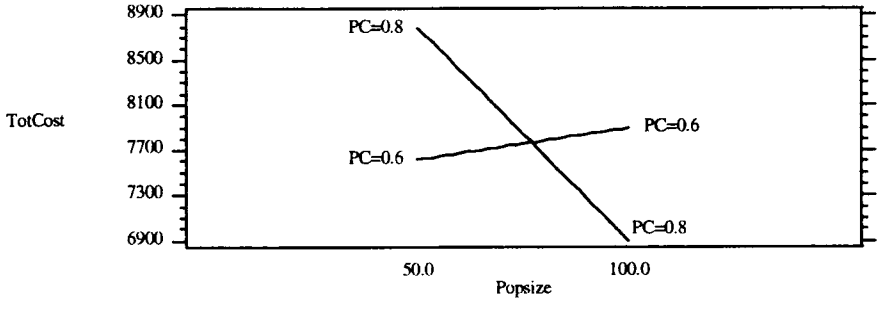


Figure B19 - Estimated Response Surface

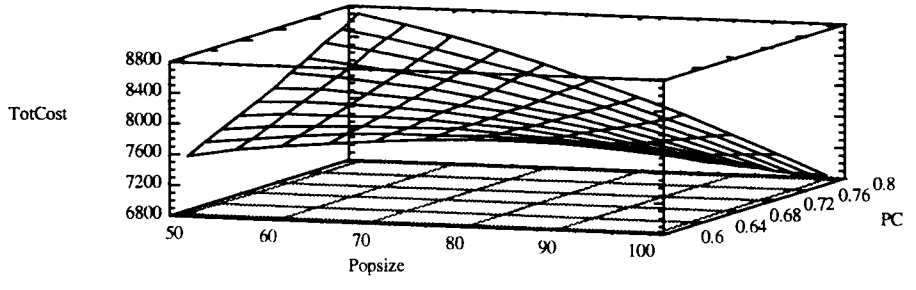


Figure B20 - Normal Probability Plot for Residuals - CompTime Heuristic 2

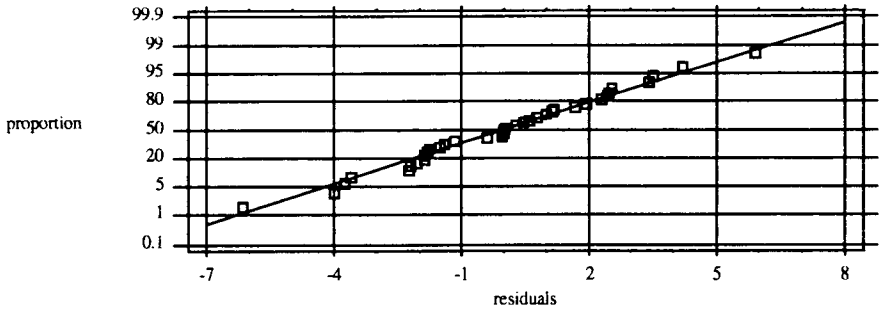


Figure B21 - Residual Plot for CompTime Heuristic 2

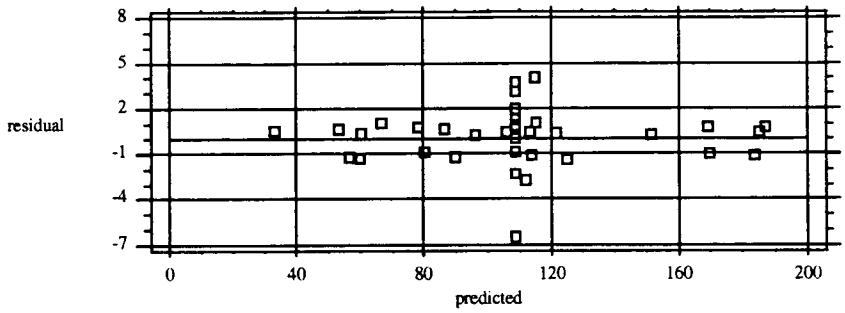


Figure B22 - Main Effects Plot for CompTime

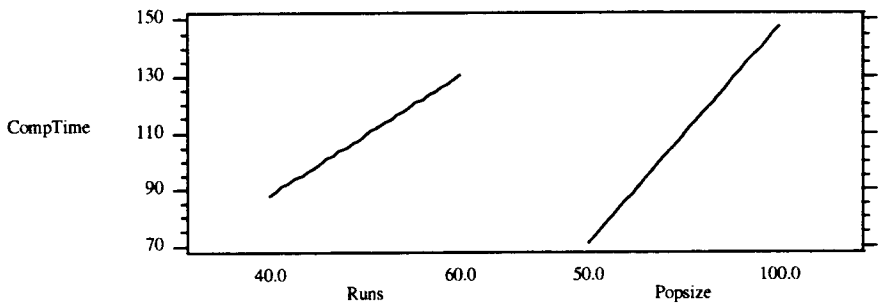


Figure B23 - Main Effects Plot for CompTime

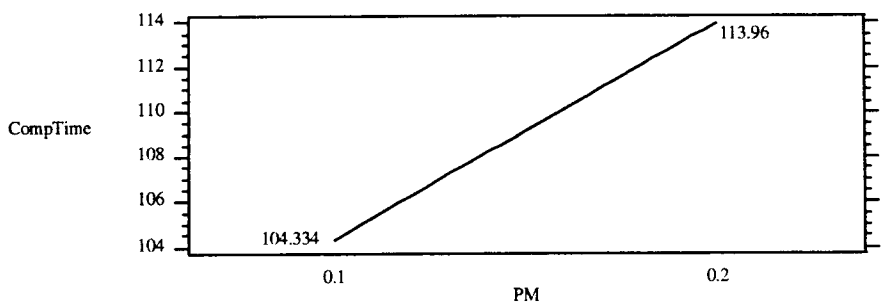


Figure B24 - Interaction Plot for CompTime

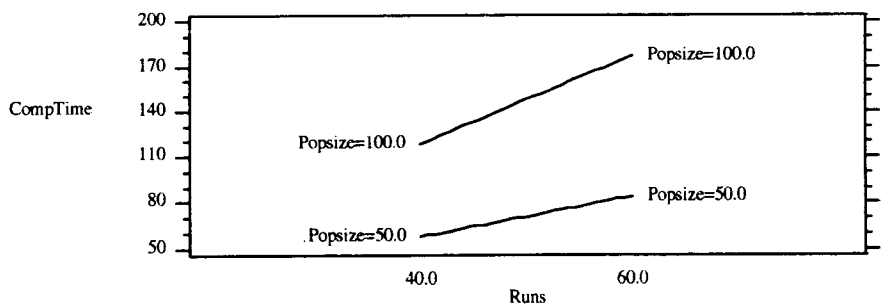


Figure B25 - Estimated Response Surface

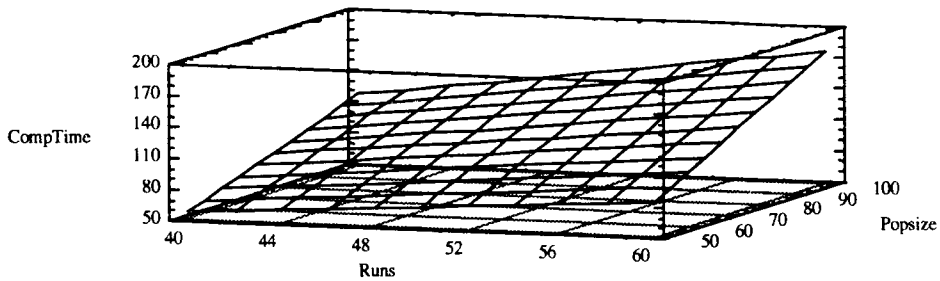


Figure B26 - Normal Probability Plot for Residuals - TotCost Heuristic 3

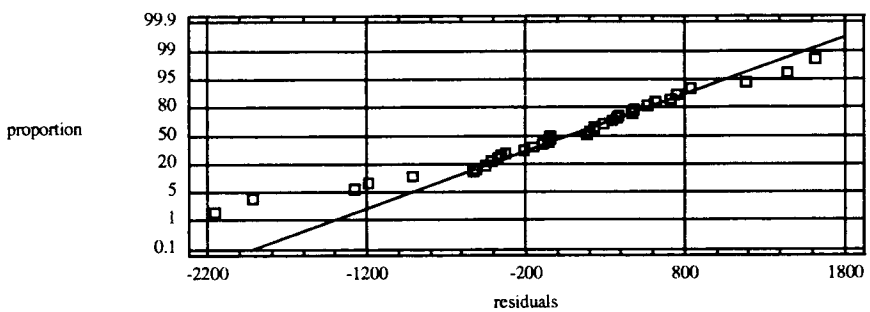


Figure B27 - Residual Plot for TotCost Heuristic 3

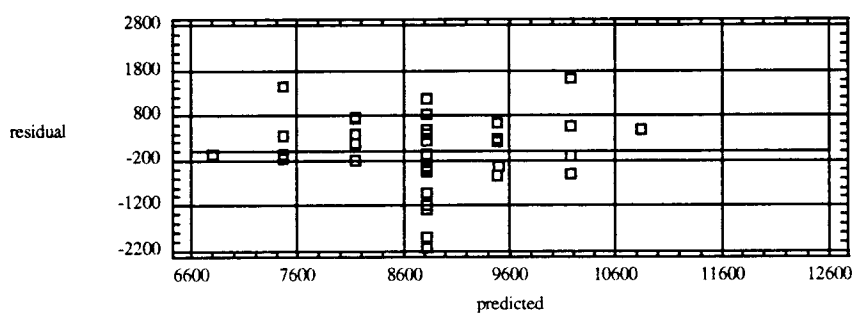


Figure B28 - Main Effects Plot for TotCost - Heuristic 3

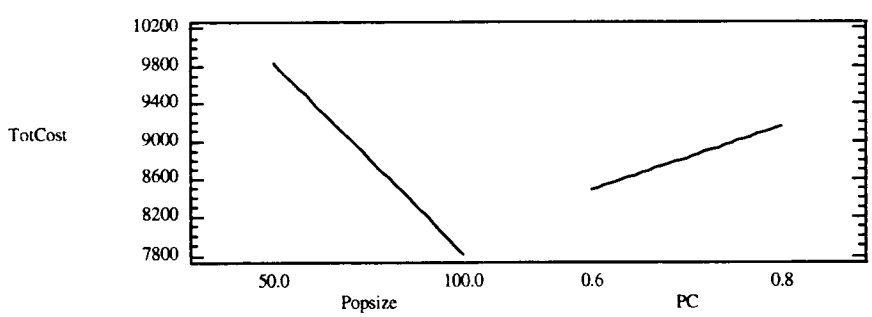


Figure B29 - Estimated Response Surface - Heuristic 3

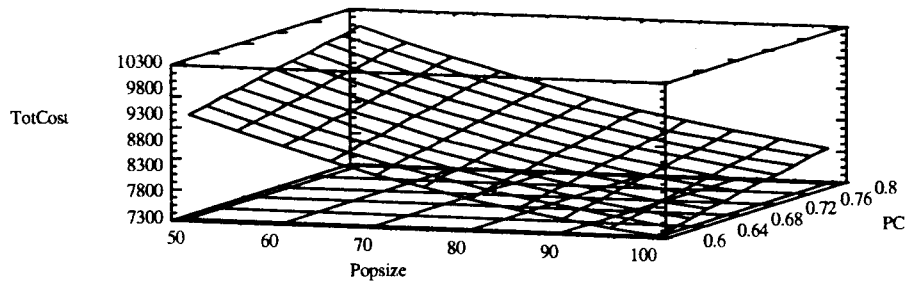


Figure B30 - Estimated Response Surface - Heuristic 3

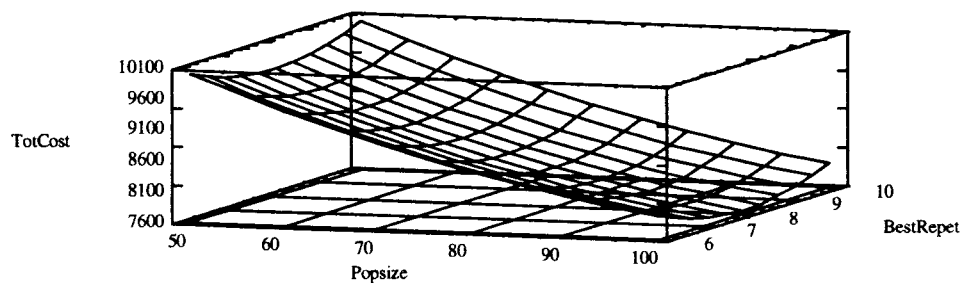


Figure B31 - Estimated Response Surface - Heuristic 3

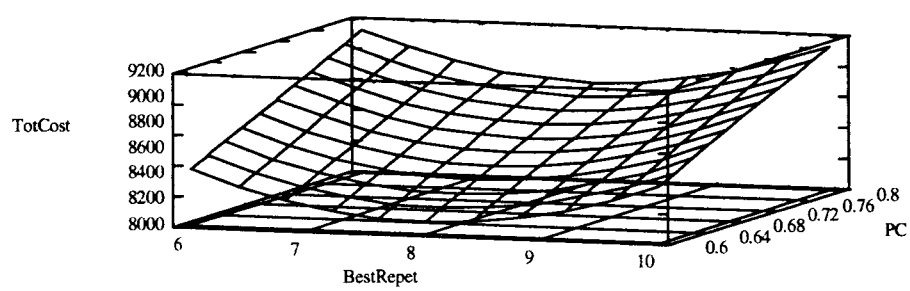


Figure B32 - Normal Probability Plot for Residuals - CompTime Heuristic 3

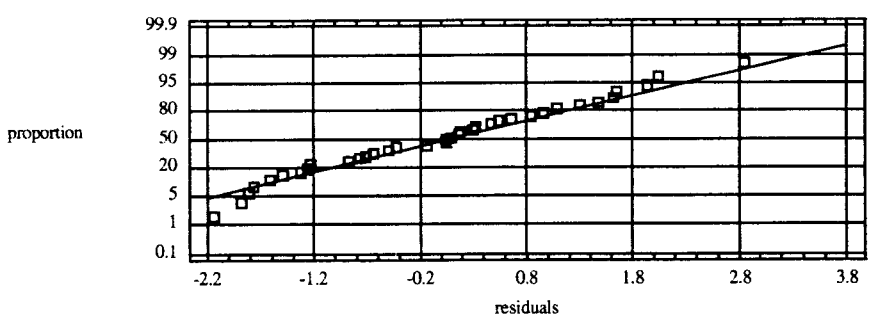


Figure B33 - Residual Plot for CompTime Heuristic 3

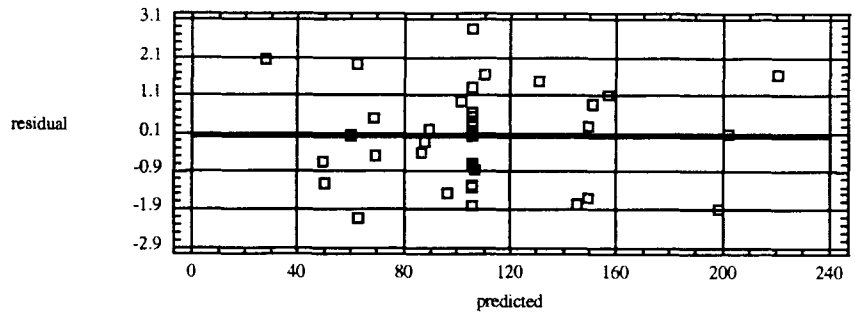


Figure B34 - Main Effects Plot for CompTime - Heuristic 3

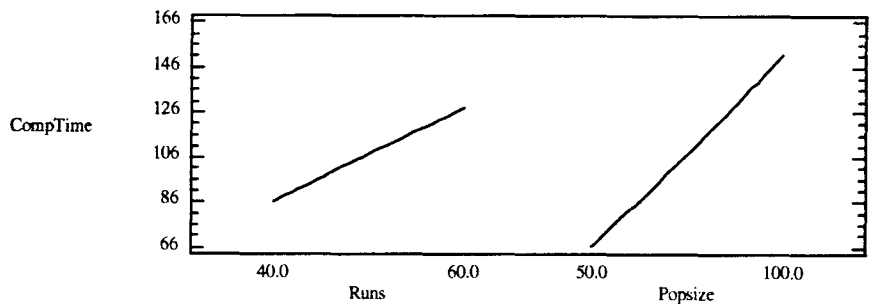


Figure B35 - Main Effects Plot for CompTime - Heuristic 3

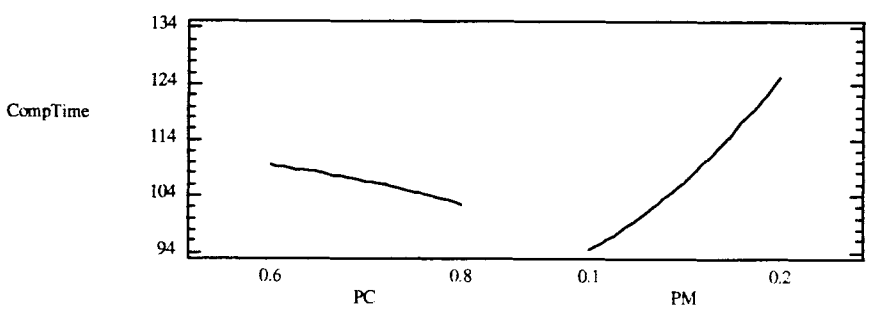


Figure B36 - Interaction Plot for CompTime - Heuristic 3

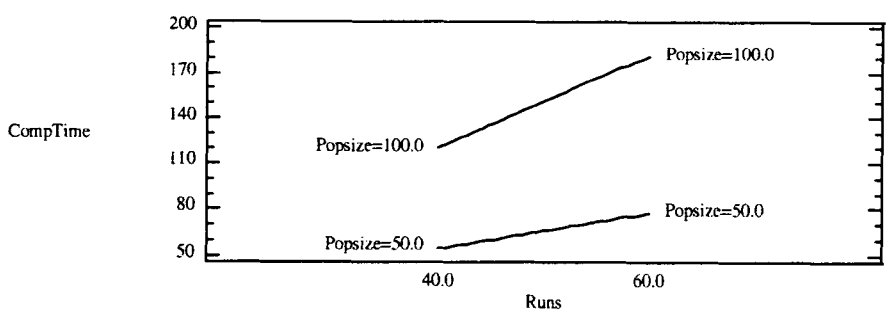


Figure B37 - Interaction Plot for CompTime - Heuristic 3

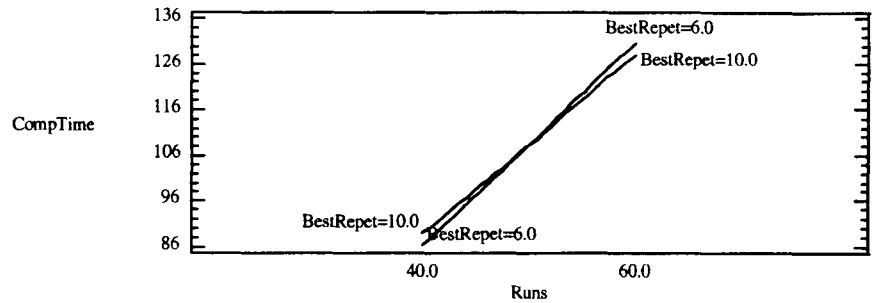


Figure B38 - Interaction Plot for CompTime - Heuristic 3

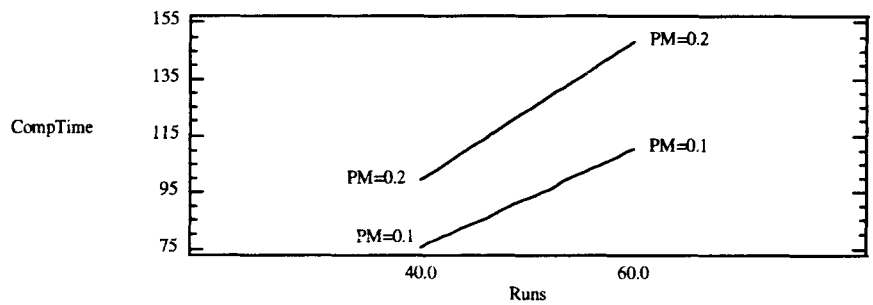


Figure B39 - Interaction Plot for CompTime - Heuristic 3

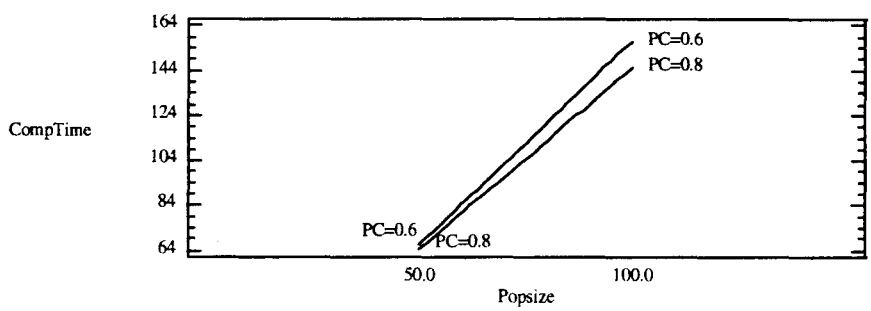


Figure B40 - Interaction Plot for CompTime - Heuristic 3

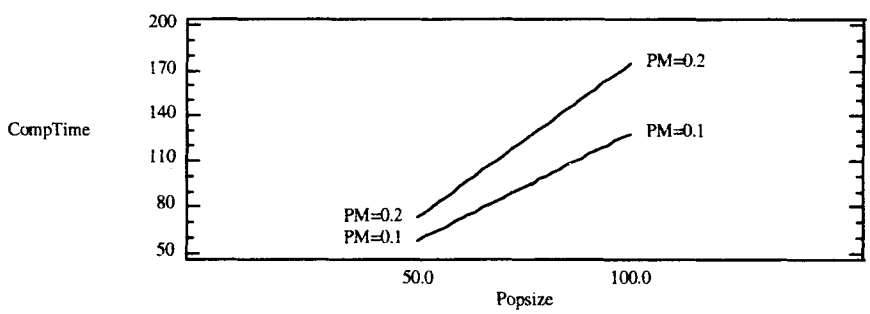


Figure B41 - Interaction Plot for CompTime - Heuristic 3

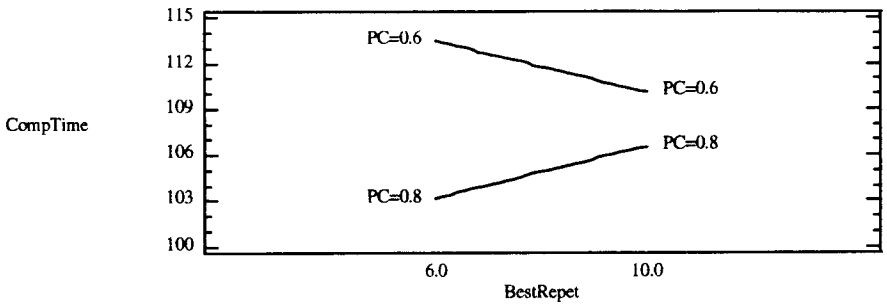


Figure B42 - Interaction Plot for CompTime - Heuristic 3

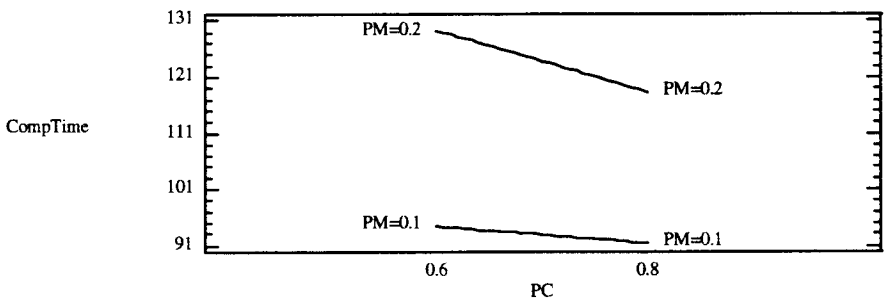


Figure B43 - Estimated Response Surface - Heuristic 3

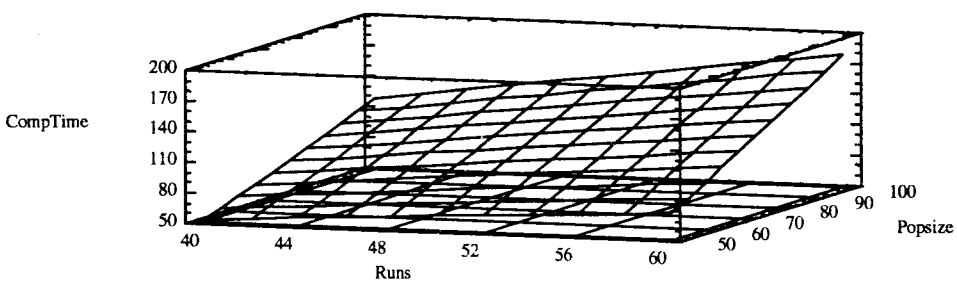


Figure B44 - Estimated Response Surface - Heuristic 3

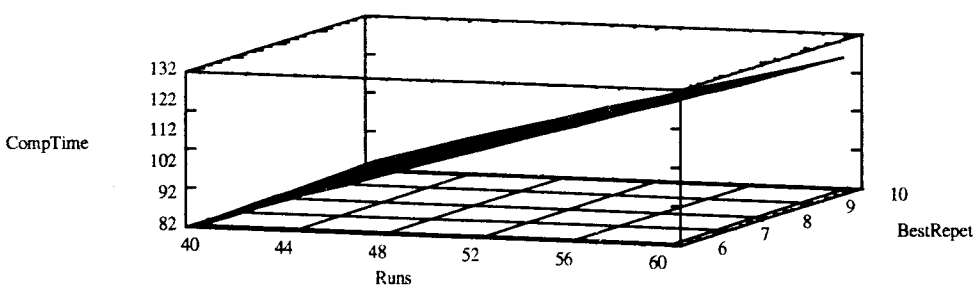


Figure B45 - Estimated Response Surface - Heuristic 3

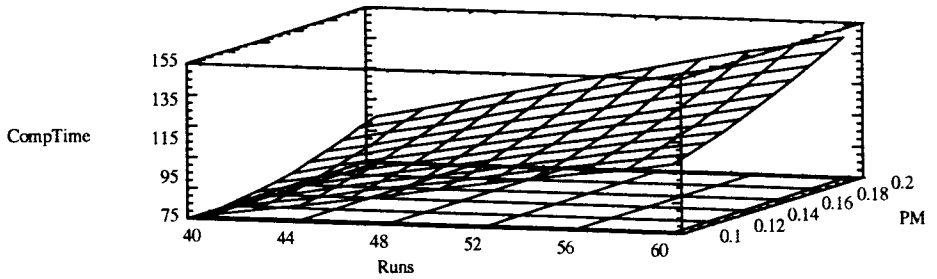


Figure B46 - Estimated Response Surface - Heuristic 3

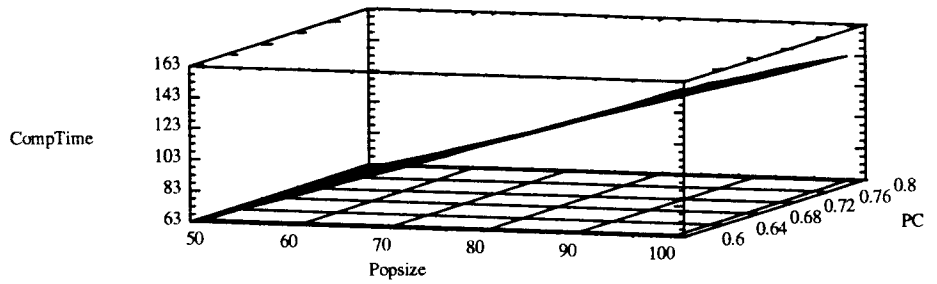


Figure B47 - Estimated Response Surface - Heuristic 3

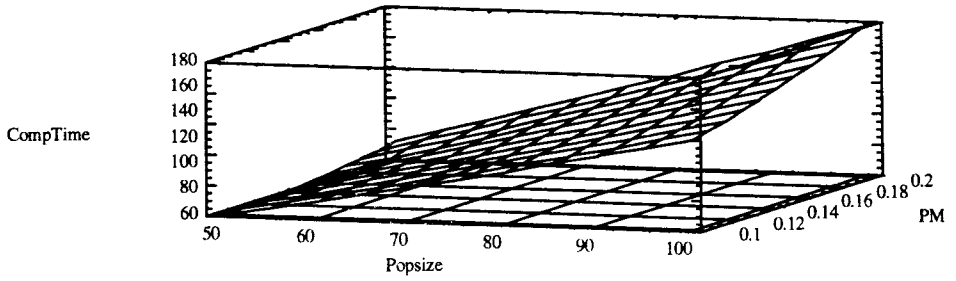


Figure B48 - Estimated Response Surface - Heuristic 3

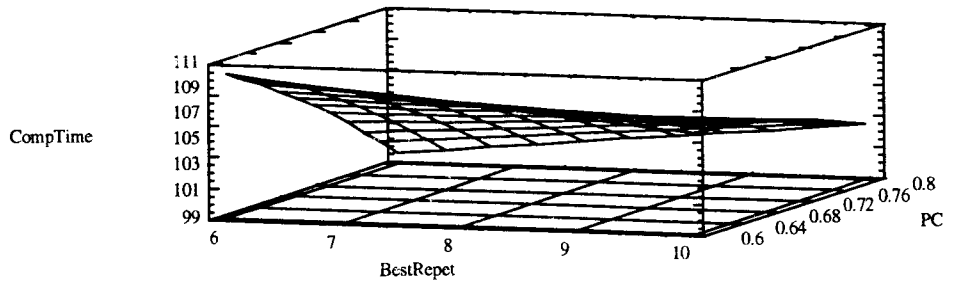


Figure B45 - Estimated Response Surface - Heuristic 3