METAHEURISTICS FOR SINGLE AND MULTIPLE OBJECTIVES PRODUCTION SCHEDULING FOR THE CAPITAL GOODS INDUSTRY

A thesis submitted for the degree of

Doctor of Philosophy

by

Wenbin Xie

Newcastle University Business School

Newcastle University

September 2011

Abstract

In the capital goods industry, companies produce plant and machinery that is used to produce consumer products or commodities such as electricity or gas. Typical products produced in these companies include steam turbines, large boilers and oil rigs. Scheduling of these products is difficult due to the complexity of the product structure, which involves many levels of assembly and long complex routings of many operations which are operated in multiple machines. There are also many scheduling constraints such as machine capacity as well as operation and assembly precedence relationships.

Products manufactured in the capital goods industry are usually highly customised in order to meet specific customer requirements. Delivery performance is a particularly important aspect of customer service and it is common for contracts to include severe penalties for late deliveries. Holding costs are incurred if items are completed before the due date. Effective planning and inventory control are important to ensure that products are delivered on time and that inventory costs are minimised. Capital goods companies also give priority to resource utilisation to ensure production efficiency. In practice there are tradeoffs between achieving on time delivery, minimising inventory costs whilst simultaneously maximising resource utilisation.

Most production scheduling research has focused on job-shops or flow-shops which ignored assembly relationships. There is a limited literature that has focused on assembly production. However, production scheduling in capital goods industry is a combination of component manufacturing (using jobbing, batch and flow processes), assembly and construction. Some components have complex operations and routings. The product structures for major products are usually complex and deep. A practical scheduling tool not only needs to solve some extremely large scheduling problems, but also needs to solve these problems within a realistic time. Multiple objectives are usually encountered in production scheduling in the capital goods industry. Most literature has focused on minimisation of total flow time, or makespan and earliness and tardiness of jobs. In the capital goods industry, inventory costs, delivery performance and machine utilisation are crucial competitive. This research develops a scheduling tool that can successfully optimise these criteria simultaneously within a realistic time.

The aim of this research was firstly to develop the Enhanced Single-Objective Genetic Algorithm Scheduling Tool (ESOGAST) to make it suitable for solving very large production scheduling problems in capital goods industry within a realistic time. This tool aimed to minimise the combination of earliness and lateness penalties caused by early or late completion of items. The tool was compared with previous approaches in literature and was proved superior in terms of the solution quality and the computational time. Secondly, this research developed a Multi-Objective Genetic Algorithm Scheduling Tool (MOGAST) that was based upon the development of ESOGAST but was able to solve scheduling problems with multiple objectives. The objectives of this tool were to optimise delivery performance, minimise inventory costs, and maximise resource utilisation simultaneously. Thirdly, this research developed an Artificial Immune System Scheduling Tool (AISST) that achieved the same objective of the ESOGAST. The performances of both tools were compared and analysed. Results showed that AISST performs better than ESOGAST on relatively small scheduling problems, but the computation time required by the AISST was several times longer. However ESOGAST performed better than the AISST for larger problems. Optimum configurations were identified in a series of experiments that conducted for each tool. The most efficient configuration was also successfully applied for each tool to solve the full size problem and all three tools achieved satisfactory results.

Dedication

I would love to dedicate this thesis to my son, Daniel Puyu Xie, who was born during my PhD research.

Acknowledgements

There are a number of people without whom this thesis might not have been written, and to whom I am greatly indebted.

To my supervisor, Professor Christian Hicks, who gave me continuous supervision, patience, trust and guidance throughout my PhD. A very special thank to Dr. Pupong Pongcharoen, for his support and valuable advice during the most difficult time and helped me overcome some bottleneck problems. I also would like to thank Dr. Tom McGovern for his guidance for my study. I am also grateful to Professor Dongping Song and Dr. Andrew Simpson for giving me some insightful feedback for my thesis.

To my wonderful wife, who continues to give me support, who was willing to engage with the struggle, and ensuring discomfort, of having a partner who refused to accept failure, has minimum negatives for the family. A very special thank you for your extreme patience despite having a husband with bad temper; for your practical and emotional support as I added the roles of husband and then father, to compete demands of work, study and being a football fan.

To my dear son, Daniel, with whom we have been separated for 3 years since you were born. When the union of a family finally came, a glance of your smile makes all the hard work worthwhile. You are my inspiration and motivation.

I am also very grateful to my father for providing much financial support throughout my study, despite having a son who likes to argue with you all the time. To my grandma, mother, brother and my parents-in-law, a very special thank to you all for keeping this family together, despite we are thousands of miles apart.

Finally I also like to thank all my friends, PhD colleagues for listening and sharing my experience. Last but not at least, I would like to thank the members of the university stuff for their help and support.

List of Published Papers

1. Xie, W., Braiden, P. and Hicks, C. (2006) 'Scheduling complex capital goods using multi-objective Genetic Algorithm', *Postgraduate Conference*. F13, Stephenson Building, University of Newcastle upon Tyne, 3-4 April. School of Mechanical & Systems Engineering, pp. 119-132.
2. Xie, W., Hicks, C. and Pongcharoen, P. (2010) 'An enhanced single-objective genetic algorithm scheduling tool for solving very large scheduling problems in capital goods industry', *The 16th international working seminar on production economics.* Innsbruck, Austria, March 1-5. V3 pp. 517-529.
3. Xie, W., Hicks, C. and Pongcharoen, P. (2010) 'Development of an Enhanced Genetic Algorithm Scheduling Tool for Production Scheduling in the Capital Goods Companies', *international journal of production Economics* (Submitted 7th April 2010).
4. Xie, W., Hick, C. and Pongcharoen, P. (2010) 'A Multi-Objective Genetic Algorithm Scheduling Tool for Solving Complex Scheduling Problems in Capital Goods Companies', *international Journal of Production Research*. (Submitted on 16th February 2011)
5. Xie, W., Hicks, C. and Pongcharoen, P. (2010) 'Development of an Artificial Immune System Scheduling Tool for production Scheduling in Capital Goods Companies, with Comparison with Genetic Algorithm', *international Journal of Production Research*. (To be submitted soon)

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

In a very general sense, scheduling is the problem of assigning a set of tasks to a set of resources subject to a set of constraints. Bachi (1999, p1) defined scheduling as "an optimisation process by which limited resources are allocated over time among parallel and sequential activities". Baker and Trietsch (2009) stated that there are two elements of scheduling: sequencing and timing. Sequencing is arranging the order of tasks that are to be performed whilst scheduling includes specifying the sequence and timing of the tasks to be performed.

In the capital goods industry, the products are highly customised in order to meet individual customer requirements and are produced in low volume on a make-to-order (MTO) or engineer-to-order (ETO) basis (Hicks and Braiden, 2000). Hicks et al.(2000) also stated that the products in the capital goods industry contain a diversity of components. The requirements for different components may vary in term of volume, quantity, customised standard and technology. In the capital goods industry, companies design and manufacture a diverse range of products and the product structure is normally deep and complex. Products such as power stations contain thousands of parts and sub-parts; hence they have many levels of assembly. The manufacturing process may contain many operations and there are precedence relationships arising from operation and assembly sequences, which give rise to the difficulties of scheduling. An optimum schedule for capital goods coordinates the supply of components to meet assembly requirements and ensures that capacity constraints are not exceeded (Stewardson et al., 2002).

Although production scheduling is a popular topic, most of the literature has concentrated on flow-shop, job-shop and assembly shops (Dimopoulos and Zalzala, 2000). Examples of flow shop scheduling research include (Burdett and Kozan, 2000, Chen et al., 2008, Eksioglu et al., 2008, Norwicki, 1999, Onwubolu and Mutingi, 1999). Job-shop focused work includes (Watanabe et al., 2005, Mattfeld and Bierwirth, 2004, Gorczyca et al., 2004, Ponnambalam et al., 2001, Al-Hakim, 2001). Work on assembly lines includes (Celano et al., 1999, Mansouri, 2004). Also Sebaaly and Fujimoto (1996a, 1996b, 1996c, 1996d) published a number of papers that considered assembly sequence planning problems (ASPP) that tried to find an optimal sequence for assembling a product consisting of n parts. Enumerative search methods have been applied for

scheduling, such as Integer Linear Programming (Manne, 1960), Dynamic Programming (Held and Karp, 1962) and Branch and Bound (Greenberg, 1968). However these methods are only appropriate for small problems because they involve enumerative search (King and Spackis, 1980). There are several well-established metaheuristic methods have been used, such as Tabu Search (TS), Simulated Annealing (SA), and Evolution Algorithms (EAs), e.g. Genetic Algorithms. These methods find a solution by a stochastic search of the problem space and are most suitable for combinatorial optimisation problems (Nagar et al., 1995). More recently, Artificial Immune System based Algorithms (AISAs) have been used to solve combinatorial optimisation problems and there were many successful applications of AISAs that have been studied to solve production scheduling problems. However, most of this research has still focused on flow-shop or job-shop scheduling problems (Chandrasekaran et al., 2006, Coello et al., 2004, Engin and Döyen, 2004, Ge et al., 2005, Kahraman et al., 2008, Ong et al., 2005, Xu et al., 2005, Zandieh and Gholami, 2009). There is no previous research that has applied AISAs to tackle production scheduling problems in capital goods industry.

Furthermore, although there is an abundant literature relating to production scheduling problems, multiple objective production scheduling in the capital goods remains a gap in the literature. In the capital goods industry, delivery performance, inventory management and resources utilisation all play a very important role. However these are conflicting objectives in production scheduling. Pongcharoen et al. (2004) pointed out that prompt and on-time delivery is important in the capital goods industry. As a result of late delivery, it is very common that the companies pay severe penalties due to the contractual terms. The delivery performance is reflected by the delivery time of final product. In order to achieve on-time delivery, final products should be completed on time according to the customer due date. On the other hand, early completion of components, assemblies and products gives rise to inventory costs and also may cause inconvenience to the customers. If companies aim to maximise resource efficiency, the idle time of machines needs to be minimised. However, this may cause a rise of inventory due to early completion of all items by increased resources utilisation. To sum up, optimising delivery performance, minimising inventory and maximising resources utilisation are conflicting. Since these objectives are conflicting, a schedule that yields a good result for one objective may not perform well for other objectives. Hence it is crucially important that multiple criteria can be taken into account at the same time.

During the last decade, considerable research has investigated the solution of multi-objective production scheduling problems. Many multi-objective approaches have been applied to tackle these problems (Choobineh et al., 2006, Cochran et al., 2003, Loukil et al., 2005, Murata et al., 1996, Pasupathy et al., 2006, Tay and Ho, 2008). Most research on multi-objective production scheduling problems has focused on flow-shops or job-shops. The typical objectives in these works are minimisation of total flow time, makespan and earliness and tardiness of jobs. In capital goods industrial, inventory costs, delivery performance and machine utilisation are crucial factors for these companies to remain their competitiveness in the market. A practical scheduling tool is needed to successfully optimise these criteria simultaneously.

Pongcharoen (2001) proposed a Genetic Algorithm based Scheduling Tool (GAST) for production scheduling in capital goods industry. However, this tool was written in the interpreted language Tk/Tcl (Ousterhout, 1994). This limited the size of problem that could be considered. In Pongcharoen's (2001) work, 'small', 'medium' and 'large' problems were respectively studied. But those problems were all still relatively small compared to realistic industrial problems. The large problem involved 118 machining and 17 assembly operations on 17 resources and there are just 4 levels of product structure. Program written in C is more computationally efficient (Kernighan and Ritchie, 1988) than interpreted language such as Tk/Tcl. Products produced by capital goods companies are characterised by high levels of uncertainty in terms of specification, lead times, demand and the duration of processes. Other interruption of production activities such as machine breakdown, sudden illness of shop operators also occur from time to time. This dynamic environment demands any re-scheduling should be completed in a short period of time which highlights the importance of reducing scheduling time in capital goods industry. The GAST only solved problems with a single objective, which was minimising the sum of tardiness and lateness penalties due to early and late completion of components, assemblies and final products. This single objective could be extended to multi-objectives to optimise delivery performance, maximise resources utilisation and minimising inventory cost. These criteria were identified by Hicks and Braiden (2000) as important performance measurements for capital goods companies. Aytug et al. (2003) reviewed a large number of papers which applied GA based approaches to solve production scheduling problems. In the last section of their paper, they were aware of the lack of justification of how and why specific GA configurations and parameters were chosen and some critical parameters

such as population size, selection scheme, termination criteria and computation time were often not reported.

1.1 **Research Objectives**

The objectives of this research project are categorised into three groups according to the three scheduling tools that developed in this research (ESOGAST, MOGAST, and AISST):

The objectives relating to the Enhanced Single-Objective Genetic Algorithm Scheduling Tool (ESOGAST) were to:

1. Produce a program similar to Pongcharoen's GAST that is written in the compiled language C (Kernighan and Ritchie, 1988), rather than the relatively slow Tk-Tcl (Ousterhout, 1994) used by Pongchareon. The objective was to allow realistic problems to be solved within a reasonable amount of time. An Enhanced Single-objective Genetic Algorithm Scheduling Tool (ESOGAST) was developed and its specification was provided; It was necessary to improve the repair process within the GA to reduce the execution time so it was able to tackle very large and complex scheduling problems, such as the full schedule described in this research;
2. Compare the performance of ESOGAST with Pongcharoen's GAST for different problem sizes in terms of solution quality and computation time;
3. Compare the performance of ESOGAST and random search combined with a repair process;
4. Identify the optimum selection schemes and best combination of crossover and mutation operators for the ESOGAST on extra large schedule;
5. Apply ESOGAST to solve a full sized scheduling problem obtained from a collaborating company using a configuration based on previous results and to identify the importance of configuration settings for the performance of Genetic Algorithm for this problem.

The objectives relating to the Multi-Objective Genetic Algorithm Scheduling Tool (MOGAST) were to:

1. Extend the single criterion GA approach to produce a Multi-Objective Genetic Algorithm Scheduling Tool (MOGAST) that simultaneously minimises inventory costs, maximise delivery performance and resources utilisation;

2. Identify the optimum crossover and mutation probabilities of MOGAST based upon the results for solving medium, large and extra large schedules. The crossover and mutation probabilities indicates the number of chromosomes that perform crossover and mutation operation;
3. Identify the best combination of crossover and mutation operators and best selection schemes for MOGAST based on results for solving medium, large and extra large schedules;
4. Evaluate the performance of MOGAST with various values of Sigma (σ) e.g. niche size, which is the threshold of dissimilarity (sometime know as the niche radius or the distance cutoff);
5. Apply the MOGAST using the optimum GA configuration based upon previous experiments to solve a full size industrial scheduling problem using the data obtained from a collaborating capital goods company;

Objectives 2-5 were based upon experiments of the MOGAST with two objectives: minimising inventory costs and maximising delivery performance

6. Identify the conflicting relationships and tradeoffs between delivery performance, inventory cost and resource utilisation; hence justify the necessity of providing such practical tool for production scheduling in capital goods industry;
7. Apply the MOGAST using the optimum GA configuration based upon previous experiments to solve the full size production scheduling problems to optimise all three objectives and benchmark the results obtained.

Objectives 6-7 were based on experiments of the MOGAST with three objectives: minimising inventory cost, maximise delivery performance and resource utilisation.

The objectives associated with the Artificial Immune System Scheduling Tool (AISST) were to**:**

1. Describe the development of Artificial Immune System Scheduling Tool (AISST) for solving production scheduling problems encountered in capital goods companies;
2. Conduct a series of experiments that were based upon a full factorial design to find the best configuration for AISST on various sizes of problems;
3. Benchmark and compare the performance of the AISST and ESOGAST algorithms in terms of the quality of solutions on same amount of search and the

computation time required;

4. Analyse the results obtained from the AISST and ESOGAST experiments.

1.2 **Outline of the Thesis**

Chapter 2 reviews the literature on various topics including the capital goods industry, their product structure and characteristics and the nature of production scheduling with conflicting multiple objectives within the industry. Scheduling and the classification of search algorithms are also explored. Genetic Algorithms and multi-objective Genetic Algorithms and Artificial Immune System are also reviewed;

Chapter 3 presents the development of the Enhanced Single-Objective Genetic Algorithms Scheduling Tool (ESOGAST). The specification of ESOGAST is described step by step and a program flowchart is also provided.

Chapter 4 describes the application of ESOGAST to solve various types of industrial scheduling problems and presents a series of experiments designed to identify the best genetic algorithms configuration in terms of operators (crossover, mutation, and selection scheme). The advantages of the ESOGAST are presented by comparing results with those obtained by GAST. The full schedule (which is extremely large) obtained from a collaborating company was solved using the optimal configuration based on results from previous experiments.

Chapter 5 presents the development of the Multi-Objective Genetic Algorithms Scheduling Tool (MOGAST). The specification of MOGAST is described step by step and a program flowchart is provided.

Chapter 6 describes the application of MOGAST to solve various types of scheduling problems with multi objectives from the collaborating company and also presents a series of experiments that were designed to identify the best multi objective genetic algorithms configuration in terms of operators (crossover, mutation, and selection scheme), parameters (population size, number of generation, crossover and mutation probability). The importance of the niche size (σ_{share}) is evaluated and described;

Chapter 7 presents the development of the Artificial Immune System Scheduling Tool (AISST). The specification of AISST is described step by step and a program flowchart

is also provided.

Chapter 8 presents experiment A that was designed to identify optimum AIS configuration in terms of number of search, percentage of antibody elimination. Experiment B was designed to analysis the results obtained and determine how to choose appropriate parameters. Experiment C compared the results obtained from ESOGAST and AISST for medium, large and extra large schedules and benchmarked their performances. In experiment D, the AISST was applied to solve the full size schedule and the results were compared with that of ESOGAST.

Chapter 9 presents the conclusions of this research. At the end, this chapter provides suggestions for further research work.

Chapter 2. Literature Review

This chapter provides the background on the capital goods industry, the characteristics of the collaborating capital goods companies, production scheduling in these companies and a classification and review of search techniques. An overview of Genetic Algorithms is provided. This chapter also presents a description of Multi-Objective Evolutionary Algorithms (MOEAs). In particular, descriptions of Vector Evaluated Genetic Algorithm (VEGA), Multi-Objective Genetic Algorithm (MOGA) and Non-dominated Sorting Genetic Algorithm (NSGA) are provided. In this chapter a brief review of Artificial Immune System inspired Algorithms (AISAs) is also presented. Finally, the research gap is clearly identified.

2.1 The Capital Goods Industry

The following section provides the definition of capital goods and describes the characteristics of the complex products in the capital goods industry.

2.1.1 What are capital goods?

Capital goods are defined as "goods (e.g. ships, railways, machinery, etc.) used in producing other goods" (Hornby, 2005, p200-201). Capital goods have played an important role in capital formation and industrialisation process in the leading industrial countries in 19th century and for those countries which started to industrialise after World War II, such as Japan and Soviet Union (Brundenius, 1987). The capital goods sector is strongly linked with the rest of the economy and has acted as a decisive instrument for the generation and diffusion of technological change (Brundenius, 1987). Rosenberg (2003) also noted that the capital goods sector has been playing a crucial role in the process of technological innovation. He stated that all these innovations, whether they introduce a new product or a better way on producing an existing product, require that capital goods sector shall produce a new product (machine) according to certain specifications.

2.1.2 Product structure and characteristics of capital goods

Welp et al (1999) stated that the product structure describes the kind, the number and the relationships between parts and assemblies. The complexity of the product structure is strongly influenced by the number of levels of assembly and the number of items within the product and assemblies. In the capital goods industry, products such as

turbine generators and oil platforms are highly customised with multiple levels of product structure. The product structure of a relatively simple product is shown in Figure 2-1.

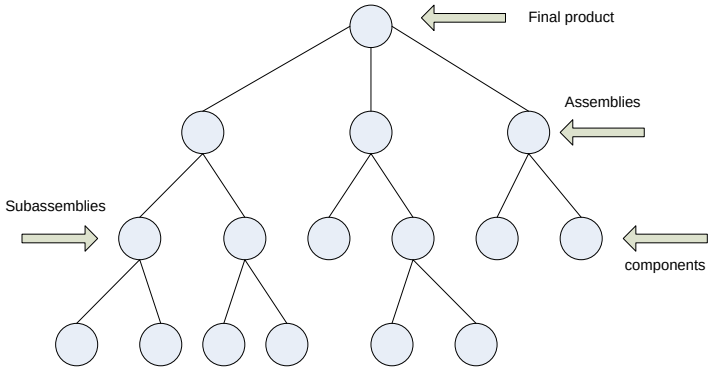

Figure 2-1 Typical simple product structure

The root node represents the final product; the leaf nodes at the bottom level represent components and the leaf nodes between represent assemblies and subassemblies respectively. A complex product may have a deep and complex structure. Hence the manufacture process of the product requires many assemblies and subassemblies. In capital goods industry, a production schedule can have a very large number of feasible sequences and this number increases exponentially with the increase of the number of items (Gottipolu and Ghosh, 2003).

Hicks et al. (2000) stated that in the capital goods industry, the requirements of different components of products may vary in term of volume, quantity, customisation/ standardisation and technology. The process routings for components are often long and may require many different types of operations on many machines. Each operation may also consist of many different activities with various time durations, such as set-up, machining and transfer time. The characteristics of components and the layout of the machines contribute to the length of set-up and transfer time respectively (Pongcharoen et al., 2004). Capacity and multiple resources constraints, complex operations and assembly precedence relationships make scheduling capital goods very difficult. An optimum schedule in capital goods industry not only needs to co-ordinate the supply of components to meet assembly requirements but also ensure that finite capacity constraints are not exceeded (Stewardson et al., 2002).

2.1.3 Description of collaborating companies

Hicks (2000) investigated four typical capital goods companies that produced capital

goods for the power generation, power distribution, materials handling and offshore industries. Hicks (1998) surveyed these collaborating companies through a series of nine visits when he was doing his research in Newcastle University. These companies were mainly engaged in make-to-order (MTO)/engineer-to-order (ETO) manufacturing and produced highly customised major products with a large design content requirement. These capital goods companies produced products such as large steam turbine generators, power station boilers, transformers and switch gear. These products had deep and complex structures and contained diverse components with many levels of assembly processes. Their companies' markets were mature with supply exceeding demand and faster and more reliable delivery are highly required by their customers (Hicks et al., 2000). There was a general shift of demand from specific items of plant toward turnkey contracts and through-life solutions (Hicks and Braiden, 2000).

There were two stages of contact with customers and suppliers that were described by Hicks (1998). In the first stage, tendering, the knowledge of the specific product application and the individual customer's preference were required for the development of the specification. Deliver performance was considered to be essential at this stage. The negotiation at this stage not only aimed to meet customer and market requirements, but also aimed to match overall project costs and lead times. In the second stage of contact, the development of an overall project plan and detailed design were involved. A series of activities such as procurement, component manufacturing, assembly, construction and commissioning then were followed.

In a competitive market product performance, delivery time, operating costs and product price were all considered highly important. Overall, intense competition in global markets had made cost reduction essential.

The product structure of the major contract products was deep and complex, while it was shallow for the sub-contract products and spares contract products. For example, thrust bearings for marine applications had a comparatively shallow product structure, whilst products like turbo-generators had a very deep product structure. However, both were produced on a MTO/ETO basis. The complex product structure with a large number of operations and assemblies made scheduling of these products very difficult. Not to mention that there are many operation and assembly precedence relationships as well as finite capacity. All this required the whole scheduling process to be strategically

planed. The lead time for the major product was several years long and in some cases exceeded the required cumulative lead time which led to delivery problems. There were some companies engaged with a spares business associated with the major products and there were also some other additional business undertaken and some companies were even doing some sub-contract products to increase the utilisation of resources.

Hicks (1998) also found that these companies had large and complex manufacturing systems. He described them as the following: There were different technologies such as casting, forging, metal cutting, fabrication and assembly. There was a large number of work centre involves processes such as jobbing, batch, flow and assembly processes. There was a various range of machine tools of both manual and computer controlled. Most of the layouts of the facilities were functional, however there was a manufacturing cell employed at one site. Some large multi-function machine tools were used in some companies and the lead times were shorten by simplifying routings and reducing set-up and process time. The assembly time was also shortened through improvement of dimensional accuracy. All these companies had a high level of capital and were highly dependent on suppliers due to large number of bought in items. The level of inventory was also very high (Hicks, 1998).

Hicks (1998) categorised these companied into two types A and B by employing a classification technique based on the work of New (1977). Company type A and B were found to have similar mix of processes and products. Company type B was found to have a number of additional mini-businesses. This was developed when demand for the major product was low and hence aimed to balance the load on the manufacturing facility. With company type A with the major products, due date performance, capital cost and cost of ownership proved essential. However regarding to the spares business, deliver performance was the key factor.

2.2 Production Scheduling in Manufacturing Systems with Product Structure

In production scheduling, the majority of the research work focused on flow-shop, job-shop scheduling where assembly activity is not considered and jobs are independent. There is also a considerable work in the literature of assembly systems which focused on coordination of assembly and sub-assembly. The relevant literature was described in page 1 of Chapter 1. However, with regards to production scheduling in manufacturing systems with a complex product structure, there is limited work. These scheduling

problems distinct themselves from the flow-shop, job-shop and assembly system is that these problems could be a combination of job-shop and assembly shop. Many components on the bottom level of the product structure require many operations to be performed on multiple machines with consideration of finite capacity and resource constraints in which is a job-shop scheduling process. The components then undergo sub-assembly and assembly operations to be assembled together to produce the final product.

Fry et al. (1989) studied the effect of product structure and sequencing rule on assembly shop performance, however they did not consider the specification of operation start times. Kim and Kim (1996) used Simulated Annealing and Genetic Algorithm for scheduling products with multi-level product structure. The product structure that was described in their work is similar to the one that were described in this research. It was assumed that all activities took place in either an assembly shop or a machine shop. An aggregated model hence was proposed which contains only two resources: a machine shop and an assembly shop. This assumption was rather inappropriate because all machining and assembly resources have different capabilities. Pongcharoen et al. (2001, 2002, 2004) develop a Genetic Algorithm Scheduling Tool (GAST) for scheduling complex products with deep product structure and multiple resource constraints. The scheduling tool was to produce an optimal schedule to minimise the total penalties arising from holding cost of components and assembly and earliness or lateness cost of final product. The optimal schedule then was shown in a Gantt chart. The tool used a four stage repair process to rectify the infeasible schedules produced due to product structure, finite capacity and resource constraints. In Kim and Kim (1996) and Pongcharoen et al. (2001, 2002, 2004), the sequencing and timing of the schedule was conducted in two separate stage. Timing was assigned after the sequences of operations were decided. The timing then was calculated in accordance with these sequences and operations were assigned as soon as its resource was available etc. operations were performed as soon as possible. However, a major problem with this approach was pointed out by Song (2001) that the solution is sensitive to the deduction rule and may not even include parts of the solution space which includes the global optimum. Hicks et al. (2007) developed a heuristic and evolutionary-strategy-based methods to solve incremental and regenerative scheduling problems for dynamic scheduling for complex engineering-to-order products. The objective of these approaches was to minimise the sum of work-in-progress holding costs, product earliness and tardiness costs. All the

literature described above however only considered a single criterion performance measurement. A multiple criteria measurement was neglect in similar scheduling environment in the literature.

2.3 Production Scheduling in Capital Goods Industry

This sector provides the definition of scheduling and describes a multiple objectives nature of production scheduling in capital goods industry.

2.3.1 Definition of scheduling

Baker (1974) defined scheduling as the allocation of resources over time to perform many tasks. Later, Bagshi (1999, p1) defined scheduling is "an optimization process by which limited resources are allocated over time among parallel and sequential activities." In scheduling, if *n* tasks are to be performed on *m* processors, there are potentially $(n!)^m$ sequences (Pongcharoen, 2001). Here it is necessary to make a distinction between scheduling and sequencing. Sequencing only concerns the ordering of operations on a single resource, while scheduling is to assign sets of sequences on resources and meanwhile simultaneously to allocate timings for operations in each resource (Bagchi, 1999).

2.3.2 Difficulties of production scheduling in capital goods industry

Products produced in capital goods companies are different from products produced in other manufacturing companies. These companies produce plant or machinery that is used to produce consumer products or commodities. Steam turbines, large boilers and oil rigs are typical products produced by capital goods companies (Xie et al., 2010). These products have their unique characteristics in production scheduling: products have a very complex and deep product structure which mainly represented by many levels of assembly. These assemblies have precedence relationships and only should be assembled from bottom to top level. These precedence relationships among assemblies may give rise to a large number of illegible schedules. At the bottom level of product structure, the manufacturing processes of these components have very long routings which are consist of many operations that operated on multiple machines with finite capacity, which may result in a large number of deadlock schedules. To sum up, it is very difficult to produce a feasible optimum schedule that can coordinate all the above requirements meanwhile optimise objectives according to decision maker's preference. Furthermore, these scheduling problems are NP-complete and the number of solutions

grows exponentially as problem size grows. For example, the full schedule problem in this work, it equals 1017 jobs operated on 36 machines. That means the number of potential solutions can be up to $(1017!)^{36}$. This gives quite a challenge to computers with modern computational power. The industry requires a very efficient and practical tool that can find optimal solutions within a realistic time.

2.3.3 Multiple objectives nature of production scheduling in capital goods industry

Many problems have multiple objectives, such as to maximise performance and minimise cost. A solution that simultaneously maximises (or minimises) all the objectives simultaneously is known as a superior solution (Evans, 1984). However, the objectives are often conflicting, preventing the simultaneous optimisation of all the objectives. One approach is to combine the individual objectives into a composite function with weightings that satisfy the decision maker's preferences. This approach produces a single solution rather than a set of solutions that can be examined for tradeoffs. In a typical problem there is a set of solutions within the search space that are superior to other solutions when all the objectives are considered, but they may be inferior to other solutions in one or more objectives. These efficient solutions are known as Pareto-optimal or non-dominated solutions. The others are dominated solutions (Srinivas and Deb, 1994). So an alternative approach is to determine a Pareto optimal solution set that produces a set of efficient solutions that are non-dominated with respect to each other. An efficient solution is one for which there does not exist another feasible solution which does at least as well on every single objective, and better on at least one objective (Evans, 1984). Moving from one solution to another involves some sacrifice in one objective to achieve some benefit in another (Konak et al., 2006).

Production scheduling in the capital goods industry is a multi-objective decision making problem. Maximising delivery performance, minimising inventory costs and maximise resource utilisation are desirable, but often conflicting objectives. It is very common for capital goods companies to be liable to pay severe penalties if products are delivered late. However, the early completion of components, assemblies and products gives rise to the stock holding costs (Song et al., 2002). Companies also aim to increase productivity and efficiency of machines. However, this also gives rise to inventory as increased machine utilisation may result in early completion of items. A tool is needed that is able to produce a set of Pareto optimal solutions, so that the decision maker can choose the solution that provides the best compromise in a particular situation.

2.4 Search Algorithms

This section describes the literature of search algorithms. These search algorithms are categorized and each of them is briefly described.

2.4.1 Classification of search algorithms

In computer science, a search algorithm refers to an algorithm for finding an item with specified properties among a collection of items within all possible items' located places or search space. In production scheduling problems, a search algorithm is used to find one or more desirable schedules that satisfy finder's requirements. Most of the production scheduling problems are NP-complete and are combinatorial optimisation problems. A great variety of optimization search algorithms has been used to solve complex combinatorial optimization problems in engineering, economics and social sciences. The objectives of optimisation can be to minimise costs, times, risk, or maximum of the quality, profits or efficiency and so on (Torn and Zilinskas, 1989). The optimization search algorithms are categorized in this research as shown in Figure 2-2.

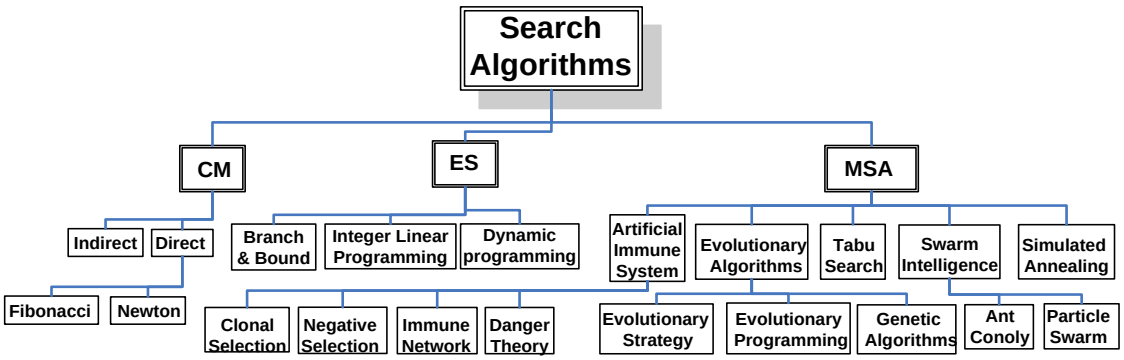

Figure 2-2. Classification of search algorithms

Key: Calculus-based Method (CM); Enumerative Scheme (ES); Metaheuristic Search Algorithm (MSA)

Calculus-based Method (CM):

Calculus-based methods are also called "strong methods" (Li et al., 2002). 'Strong methods' are those with a strong mathematical foundation that rely heavily on problem-specific knowledge whereas 'weak methods' are those without a strong mathematical foundation that require little or no problem-specific knowledge, such as Genetic Algorithms (Goldberg, 1989). These methods have been widely studied and may be subdivided into two main classes: direct and indirect methods. Direct methods, such as hill climbing, is a direct process of making small changes to a putative solution until no

further improvement is possible. Direct methods are methods that do not require any information about the gradient of the objective function, while the indirect methods mostly are those gradient based methods (Li et al., 2002). The indirect methods aim to find the local optimum by solving the nonlinear set of equations resulting from setting the gradient of the objective function equal to zero (Goldberg, 1989). The first shortcoming of these methods is that they are local in scope. The zero-finding procedures would lead us to miss another peak point. The second is that they depend upon the existence of derivatives (Goldberg, 1989). These methods also require a mathematical model that represents the problem, which is difficult or impossible to produce for combinatorial optimisation problems.

Enumerative Scheme (ES):

The idea of enumerative schemes is quite simple and straightforward: looking at the objective function values at every point in the search space, one at a time. This method is a very human kind of search (Goldberg, 1989), but there is a severe shortcoming: lack of efficiency. For example, for the early research, methods such as Integer Linear Programming (Manne, 1960), Branch and Bound (Greenberg, 1968) and Dynamic Programming (Held and Karp, 1962) are all enumerative search methods that used to solve optimisation problems. Although these methods can find the optimal point in the whole search space, many practical spaces are too large to search fully.

Metaheuristic Search Algorithm (MSA):

Metaheuristics solve an optimisation problem, with the existence acknowledge of a solution, by improving this solution iteratively, with regard to this solution's quality using a measurement procedure. Metaheuristic search algorithms are flexible methods for solving complex combinatorial optimization problems in which other methods may not be applicable. They can find a very good solution without needing a lot of pre-experience information relating to the problem. These methods use random choice as a tool to guide the search and they can be adapted to the optimization problems in different domains and normally a satisfactory solution can be obtained. Metaheuristic search algorithms are also called the 'weak' methods (Li et al., 2002). There were many metaheuristic search algorithms that have been continuously proposed in literature and Appendix A lists the main metaheuristic search algorithms and the significant contributions along its history. The popular methods are Simulated Annealing (Kirkpatrick et al., 1983), Taboo Search (Glover, 1989) and Genetic Algorithms

(Holland, 1975) and more recently Ant Colony Optimisation Algorithms (Dorigo et al., 1996), Particle Swarm Optimisation Algorithm (Eberhart and Kennedy, 1995) and Artificial Immune System based Algorithms that inspired by the immune theory of Clonal Selection (Burnet, 1959), Negative Selection (Lederberg, 1959), Immune Network (Jerne, 1974) and Danger theory (Matzinger, 1994). These methods have been intensively studied and applied to solve real-world optimisation problems.

Among all the Metaheuristics described above, Genetic Algorithms appeared to be very popular and have been intensively studied and applied to many real-world combinatorial problems (Aytug et al., 2003, Coello, 2000, Dimopoulos and Zalzala, 2000). Hence a description of Genetic Algorithms was provided in the following section.

2.5 An Overview of Genetic Algorithms

This section describes elements of Genetic Algorithm and how it works. It also states the advantages of GA against other traditional methods and a schematic of GAs are also provided.

2.5.1 Introduction: What are Genetic Algorithms?

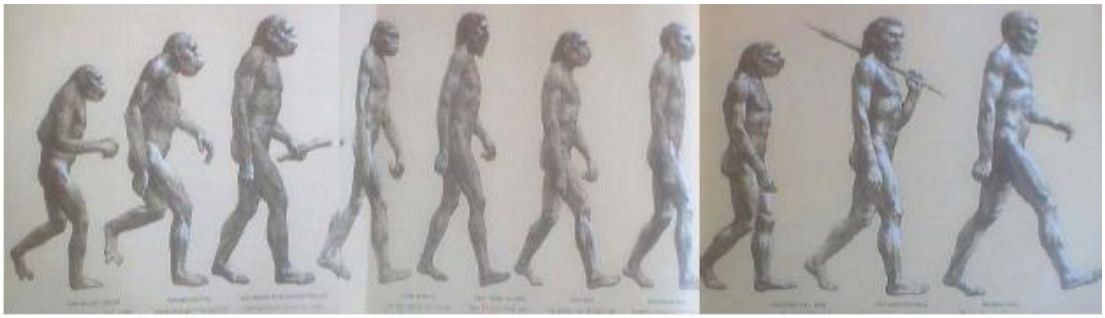

Figure 2-3. The evolution of human beings (Adra, 2003)

Based upon the principle of Darwinism, the 'survival of the fittest', only the best individuals were able to survive several biological crises and climatic breakdowns, prehistorically (Adra, 2003). Figure 2-3 shows the evolution of human beings over millions of years.

The Genetic Algorithm (GA) is a stochastic search method that mimics the metaphor of natural biological evolution (Holland, 1975). It is a search technique for approximating optimal solutions within complex search spaces (Goldberg, 1989). The GA operates on "populations" of potential solutions by applying the principle of nature selection. The

probability of a strong individual to survive is greater than a weak individual, hence GA is able to produce better individuals generation by generation until a stopping criterion is reached (e.g. certain number of generations or a mean deviation in the population) (Goldberg, 1989).

2.5.2 How Genetic Algorithms work?

Aytug et al. (2003) considered eight basic components of a Genetic Algorithm: genetic representation, initial population, evaluation function, reproduction selection scheme, genetic operators, generational selection scheme, stopping criteria and parameter setting.

The GA mechanism starts by encoding the problem into a representation of a chromosome, which made of a string of genes, so that the chromosome values are uniquely mapped onto the decision variable domain (Zalzala and Fleming, 1997). Initial populations are usually generated by combining genes randomly.

Starting from an initial population of potential solutions, the GA performs genetic operations: crossover and mutation. Crossover exchanges genetic materials between two or more parents, while mutation slightly changes an individual by mutating a gene. Crossover enables the GA to exploit the current neighbourhood of the search space and possibly leads GA to a local optimum, while mutation leads the GA to a next or far different neighbourhood of the search space (Goldberg, 1989). So crossover is called 'exploitation operator', while mutation is called 'exploration operator' (Li et al., 2002).

The evaluation function assesses the fitness of individual members of a population. The fitness values represent their likelihoods to be selected in the present environment. This means through the evaluation function, the mechanism for selection of individuals which will be selected to either mate or replace new generation is now established. There are known as reproduction selection and generational selection.

The popular selection methods to select individuals to perform genetic operations are random selection, Roulette Wheel Selection (Goldberg, 1989) and Tournament Selection (Goldberg, 1989). Roulette Wheel Selection gives individuals with better fitness a better chance to be selected. While Tournament Selection randomly selects n individuals ($n \geq 2$) to compete against each other and the fittest will be selected.

Now new individuals are created through genetic operations and this procedure will carry on until the number of these individuals equals the size of the population. Once these new chromosomes have been assigned a fitness value, they can be chosen from the population base upon their fitness by performing generational selection scheme such as Roulette Wheel (Goldberg, 1989), Tournament approach(Goldberg, 1989) and Elitist strategy (De Jong, 1975). The selected chromosomes will then form the next generation. The processes above are repeated until a termination condition is satisfied. Stopping criteria usually are the number of generations, population diversity, entropy or stop execution when the average of best population fitness has not increased in the last *t* generations (Aytug et al., 2003). A typical schematic of GAs is shown in Figure 2-4.

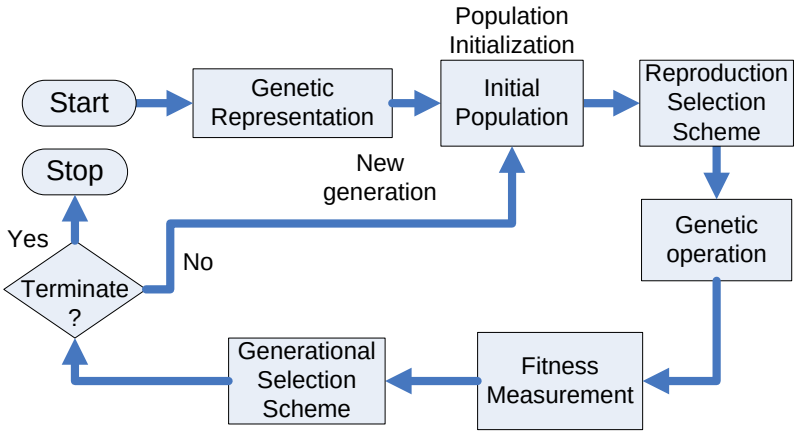

Figure 2-4. A schematic of GAs

2.5.3 Genetic Algorithms versus traditional methods

Genetic Algorithms (GAs) differ substantially from traditional search methods such as enumerative search. Based on the literature (Gen and Cheng, 1997, Goldberg, 1989, Li et al., 2002, Michalewicz, 1992, Mitchell, 1998, Rothlauf, 2006, Wang and Cao, 2002, Zalzala and Fleming, 1997, Bagchi, 1999), the differences or advantages of genetic algorithms are summarised as follows:

1) Instead of a single point, GAs search a population of points in parallel;
2) Instead of deterministic rules, GAs use probabilistic transition rules;
3) Instead of the parameter set itself, GAs work on an encoding of a parameter set;
4) GAs do not require derivative information or other auxiliary knowledge; only the objective function and corresponding fitness values influence the directions of search;
5) GAs are a self-organizing, self-adapting and self-learning algorithms with their own intelligibility;

6) GAs may be easier to be applied directly than other traditional methods;
7) GAs provide a population of potential solutions rather than a single solution.

Genetic Algorithms conduct both exploitation and exploration in their search strategies by applying crossover and mutation (Goldberg, 1989). The mutation operator, sometimes known as the 'exploration operator', tends to move the search to a new neighbourhood, whereas crossover, the 'focusing operator', helps the GA move towards a local optimum. Crossover tends to make the individuals within the population more similar, whereas mutation tends to make them more diverse (Islier, 1998). Through the genetic operation, GAs can find better solutions generation by generation (Li et al., 2002). The search mode of GAs is shown in Figure 2-5.

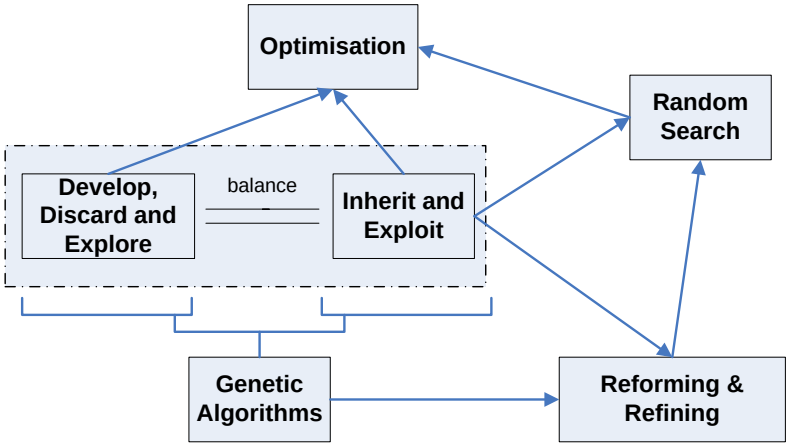

Figure 2-5. The search mode and ability of GAs (Li et al., 2002)

2.6 A Literature of Multi-Objective Evolutionary Algorithms

This section provides a review of Multi-Objective Evolutionary Algorithms. Descriptions of Vector Evaluated Genetic Algorithm, Multi-Objective Genetic Algorithm and Non-dominated Sorting Genetic Algorithm are provided.

2.6.1 Multi-objective optimization

Multi-Objective optimisation is playing a more and more important role for scientists and engineers. Most real-world problems have more than one objective. Because of a lack of suitable and efficient approaches, these problems were mostly solved as single-objective optimisation problems in the past. But there is a significant difference between the working principles of single and multi-objective optimisation algorithms (Deb, 2004). In a single objective optimization problem, the task is to find one solution

according to a single objective function. With a multi-objective optimisation problem, the task is to find optimal solutions with respect to each of the objective function. In a general multi-objective optimisation problem, the general form could be stated as follows:

$$\left.\begin{array}{ll} Max(min) & F(x) = \{f_1(x), f_2(x), \dots, f_n(x)\}; \\ Subject\ to & g_i(x) \leq b_i \quad x \in P, i = 1,2, \dots, m; \end{array}\right\} \qquad \text{Equation 2-1)}$$

In this formulation, there are *n* objective functions *f(x)* are considered in formulation *F(x)*, where $\boldsymbol{x}$ is a vector of *n*-dimensional decision variables in decision space *X* and $g_i(x)$ are certain constraints. The last set of constraints ($x \in P$) is called variable bounds, restricting each decision variable x_i to take a value within space *P*. In multi-objective optimisation, the objective functions constitute a multi-dimensional space called the objective space which maps with the decision variable space. For each solution *x* in the decision variable space, there exists a corresponding point in the objective space (Deb, 2004).

Most multi-objective optimisations use the concept of domination. A solution x_1 is said to dominate solution x_2, if solution x_1 is no worse than x_2 in all objectives (Fonseca and Fleming, 1993). Such solution x_1 is also called non-dominated solution. Most multi-objective optimisation methods use this domination concept to search non-dominated solutions(Deb, 2004). By comparing all the solutions, it will find out which solution dominates which and which solutions are non-dominated with each other, hence it is expected to find a set of solutions in which any two of them do not dominate each other. This set is called the non-dominated set for the given set of solutions and these solutions are also called Pareto-optimal solutions (Deb, 2004).

The literature relating to multiple-objective optimisation methods can be traced back at least six decades. Table B-1 in Appendix B provides a classification of Multi-Objective Optimisation Algorithm in which these algorithms in literature were categorised into two groups: classical methods and evolutionary algorithms. All of the classical algorithms tried to convert a multi-objective optimisation problem into a single-objective optimisation problem. For instance, the Weighted Sum Method (Syswerda and Palmucci, 1991) minimised a weighted sum of multiple objectives, the ε-Constraint Method (Haimes et al., 1971) optimised one objective function and considering all other

objectives as constraints. Deb (2004) stated that these methods result in a single-objective optimisation problem, which must be solved by using a single-objective optimisation algorithm. Deb (2004, p74) studied these methods and observed a number of difficulties: "1, only one Pareto-optimal solution can be expected to be found in one simulation run of a classical algorithm; 2, not all Pareto-optimal solutions can be found by some algorithms in non-convex Multi-Objective Optimisation Problems (MOOPs); 3, all algorithms require some problem knowledge, such as suitable weights in Weighted Sum Method (Syswerda and Palmucci, 1991) or constraints ε in ε-Constraint Method (Haimes et al., 1971)."

2.6.2 A brief review of Multi-Objective Evolutionary Algorithms and its applications in production scheduling

Evolutionary Algorithms have drawn much interest from researchers over the last two decades and nearly a hundred Multi-Objective Evolutionary Algorithms (MOEAs) have been introduced (including modified and developed ones) over the years to try to deal with MOOPs (Tan et al., 2005, p11). There are several reviews of multi-objectives optimisation techniques, such as works from Fonseca and Fleming (1995), Coello (2000) and Konak et al. (2006) and books of Bagchi (1999), Coello and Van Veldhuizen (2002), Deb (2004) and Tan et al. (2005). Multi-Objective Evolutionary Algorithms (MOEAs) were categorised into three groups by Deb (2004): Non elitist MOEAs, elitist MOEAs and constrained MOEAs and they are shown in Table B-1 in Appendix B. A list of descriptions of popular multi-objective genetic algorithms is also provided in Table B-2 in Appendix B based on the work done by Konak et al. (2006). The fitness assignment, diversity mechanism, advantage and disadvantage of these algorithms are described.

Evolutionary Algorithms (EAs) are stochastic search techniques that mimic evolutionary processes in nature, based upon Darwin's theory of evolution (Darwin, 1859). The population-based approach makes EAs particularly suitable for multi-objective optimisation problems. They can simultaneously search different regions of the search space and find a diverse set of solutions in discontinuous, non-convex or multi-modal solution spaces (Konak et al., 2006). A Pareto optimal set provides solutions that are non-inferior or non-dominated with respect to each other and offer different compromises between the various objectives. For solutions within the Pareto set, improving performance in terms of one objective requires some sacrifice in terms of a competing objective (Konak et al., 2006). The region represented by the Pareto set is

known as a Pareto Front (Coello, 2000).

Schaffer (1985) proposed the first multiple objective genetic algorithm named the Vector Evaluated Genetic Algorithm (VEGA) which was intended to find Pareto front solutions, where all points along the front represented solutions with different combination of equally desirable outcomes. Although this approach was partially successful, VEGA tended only to find extreme points on the Pareto front, where one attribute was maximal (Horn et al., 1994). Many new approaches and variations have since evolved. Comprehensive reviews have been published (Fonseca and Fleming, 1995, Fonseca and Fleming, 1998, Coello, 2000, Zitzler et al., 2000, Konak et al., 2006).

After VEGA was proposed, Fonseca and Fleming (1993) noted that in the VEGA, individuals will be sampled at different rates and tend to split into different species with each of them particularly strong in one of the objectives. This will lead to distinct individuals and eventually distinct species. Schaffer (1985) noticed this property of VEGA and called it speciation. Speciation is undesirable and against the aim to find optimal solutions (Fonseca and Fleming, 1993). In analogy from nature, Bachi (1999) described niche as the role that a particular population (the members of a single species) plays within the ecosystem. Within the environment, there are different subspaces (niches) that can support different types of life (species or organisms). In a multi-modal domain, each peak or optimum can be viewed as a niche (Miller and Shaw, 1996). In a GA, the number of individuals supported by a niche should be proportionate with the niche's fitness. The niches should be populated according to their fitness relative to other niches. This is known as a niche proportionate population (Deb and Goldberg, 1989). Niche formation maintains the diversity of the population hence helps GA to find a set of optimal solutions (Deb and Goldberg, 1989). Niche formation and speciation are foundations of Multi-Objective GAs (Bagchi, 1999).

It is important for multi-objective GAs to keep a diverse population in order to produce solutions that are uniformly distributed throughout the Pareto front. Unfortunately, GAs tend to suffer from a phenomenon known as 'genetic drift' which causes solutions to be produced in relatively few clusters e.g. niches (Rogers and Prugel-Bennett, 1999). Although Deb and Goldberg (1989) firstly comprehensively introduced and explained niche and speciation in genetic algorithm, Cavicchio's study (1970) was one of the first attempts to introduce niche-like behaviour in adaptive systems. This mechanism named pre-selection operator replaces a parent with an offspring if the fitness of the offspring is

better than the worst parent, hence allowing multiple important solutions to be maintained in the population. De Jong (De Jong, 1975) generalised this technique and introduced a crowding model to maintain population diversity. In De Jong's approach a fraction of population (G, the generation gap) reproduces and dies each generation. The offspring produced replaces the most similar individuals within the subpopulation of size CF (the crowding factor) (Sareni and Krahenbuhl, 1998). Goldberg and Richardson (1987) developed fitness sharing as an alternative niching approach. Instead of substituting similar solutions, their approach degraded the fitness of similar solutions to help to reduce crowding and maintain diversity. Sigma (σ), also known as niche size, is a very important parameter introduced in this sharing model.

Goldberg and Richardson's (1987) procedure is as follows. The shared fitness f_i' of an individual i with fitness f_i is calculated using equation 2-2), where m_i is the niche count.

$$f_i' = \frac{f_i}{m_i} \qquad \text{Equation 2-2)}$$

The niche count is calculated by summing a sharing function over all members of the population as shown in equation 2-3), where N is the population size and d_{ij} is the distance between individuals i and j.

$$m_i = \sum_{j=1}^{N} sh(d_{ij}) \qquad \text{Equation 2-3)}$$

The sharing function *sh* measures the similarity between pairs of individuals as shown in equation 2-4), where σ_s is the threshold of dissimilarity (sometimes known as the niche radius or the distance cutoff) and α is a parameter that determines the shape of the sharing function. Individuals within σ_s of each other degrade each others' fitness because they are within the same niche. As the niche becomes increasingly populated its niche count increases so that its shared fitness becomes lower than other niches (Horn et al., 1994).

$$sh(d_{ij}) = \begin{cases} 1 - (d_{ij}/\sigma_s)^{\alpha}, if\ d < \sigma_s \\ 0, \qquad\qquad otherwise \end{cases} \qquad \text{Equation 2-4)}$$

The use of a sharing function helps the search cover the whole search space and helps the formation of stable subpopulations. However, setting the threshold of dissimilarity σ_s requires a priori knowledge of the spacing of optima; but unfortunately the distance between optima is unknown and the peaks may not be equidistant. It therefore has to be estimated. The sharing function is also computationally expensive (Sareni and

Krahenbuhl, 1998).

Deb and Goldberg's work (1989) inspired other researchers to build up other niche-based schemes to find Pareto optimal solutions. The first outcome was the Multi-Objective Genetic Algorithm (MOGA) developed by Fonseca and Fleming (1993), which introduced a rank-based fitness assignment method and allowed the external intervention of a decision maker. The interaction between the Genetic Algorithm and the decision maker led to the determination of satisfactory solutions. This approach includes a ranking process in the search procedure which assigns all non-dominated solutions to rank 1. Other individuals are ranked with respect to the remaining population. Many individuals may be assigned to a particular rank, which is then used to select or delete blocks of points from the mating pool. This is likely to produce a significant selection pressure that leads to premature convergence (Srinivas and Deb, 1994, p10). Srinivas and Deb (1994) also found that MOGA cannot guarantee the fitness of individuals in lower level rank always better than those individuals in higher level rank due to the existence of crowding solutions. They hence proposed an approach called Non-dominated Sorting Genetic Algorithm (NSGA) to overcome this disadvantage. Inspired by the previous work such as MOGA and NSGA, many approaches were proposed to solve multiple objective optimisation problems. (see example from Sareni and Krahenbuhl, 1998, Bagchi, 1999, Coello et al., 2002, Deb, 2004, Tan et al., 2005).

The following sections describe three types of Multi-Objective Evolutionary Algorithms (MOEAs): the first multi objective GA: Vector Evaluated Genetic Algorithm (VEGA) (Schaffer, 1985); the first multi objective GA using Pareto ranking: Multi-objective Genetic Algorithm (MOGA) (Fonseca and Fleming, 1993); and the first GA using non-domination sorting for ranking: Non-dominated Sorting Genetic Algorithm (NSGA) (Srinivas and Deb, 1994). The advantages and disadvantages of these algorithms are also described.

2.6.3 Vector evaluated Genetic Algorithm (VEGA)

The Vector Evaluated Genetic Algorithm (VEGA) was proposed by Schaffer (1985) and was the first multi-objective GA to find a set of non-dominated solutions. This VEGA is a straightforward extension of a single-objective GA (Deb, 2004). It evaluates an objective vector, instead of a scalar objective function. Each of the elements of the vector represents an objective function. In order to tackle a number of objectives (m),

Schaffer (1985) divided each of the population (size *P*) into *m* subpopulations. So each of the subpopulation has *P/m* individuals and then these individuals (subpopulations) are shuffled and randomly grouped to form a new population to size *P* by performing the genetic operations. Each of the subpopulations is assigned a fitness based on a different objective function. This process is continued to the following generation until the stop criterion is met.

The main advantage of VEGA is that it is easy to implement because it was developed from a simple GA and only minor changes were needed to be made to convert it to a multi-objective GA (Deb, 2004). As described above, because each subpopulation is assigned fitness based on each objective accordingly, VEGA has a tendency to find best solutions of each objective.

However, in a VEGA, each solution is evaluated only with one objective function. Hence every solution is not tested for other objective functions. Although VEGA prefers the best solution regarding to each objective by genetic operator in each subpopulation, Schaffer (1985) observed that the diversity of solutions could not be maintained and eventually VEGA converges to individual best solutions only.

2.6.4 Multi-Objective Genetic Algorithm

The Multi-Objective Genetic Algorithm (MOGA) was introduced by Fonseca and Fleming (1993). This MOGA classified the populations -'potential solutions of the problem'- into two groups – non-dominated solutions and dominated solutions. MOGA explicitly emphasise non-dominated solutions and simultaneously maintains diversity in the non-dominated solutions by introducing niche formation among solutions of each rank by using a rank based fitness assignment. So the MOGA is based upon the similar structure as a classical GA, except the way fitness is assigned to each solution in the population is different. It ranks each individual solution according to the number of solutions in the current population by which it is dominated. In the MOGA, before assigning the rank, each solution is calculated for its domination in the population. To a solution *i* in current population, a rank equals to one plus the number of solutions that dominate solution *i*. Deb (2004) stated as:

$$r_i = 1 + n_i \qquad \text{Equation 2-5)}$$

Here r_i represents the rank of a solution *i*, n_i represents the number of solutions that

dominate solution *i*. All non-dominated solutions are assigned rank 1, since no solution would dominate a non-dominated solution in a population. (Note that, concerning fitness assignment, in any population, there must at least one solution is assigned to rank 1 and the maximum rank of any population cannot exceed the population size, *N*.) Furthermore, not all the possible ranks (between 1 and N) will necessarily be assigned in the population at a particular generation.

Figure 2-6 shows 10 solutions of a two-objective minimisation problem while Figure 2-7 shows the rank of each solution. It can be easily seen that not all the ranks are represented in a population. Here, for example, rank 7, 9 and 10 are absent.

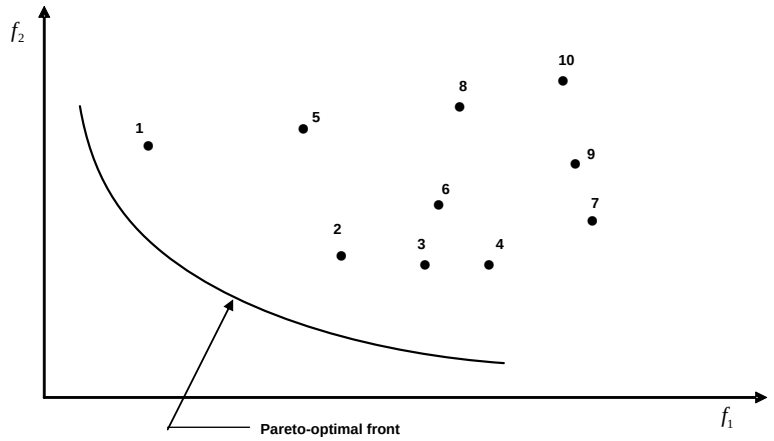

Figure 2-6 Solutions within a two-objective search space (Deb, 2004)

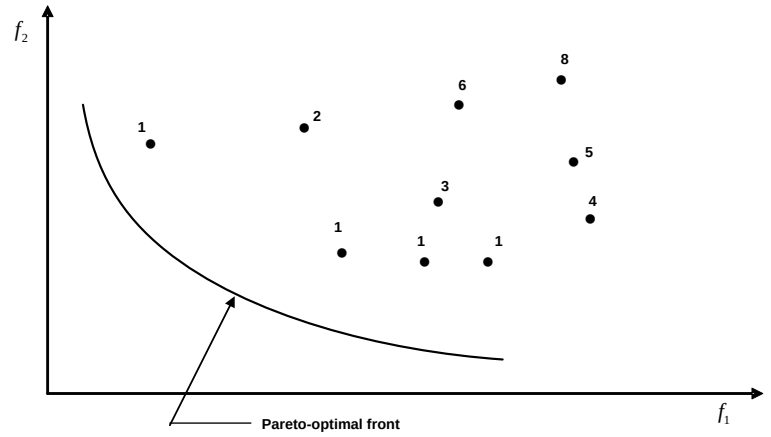

Figure 2-7 Ranked solutions within a two-objective search space (Deb, 2004)

Once all the solutions have been ranked, the fitness is assigned to individuals according to the rank. Fonseca and Fleming (1993) defined three phases to be followed:

1. Sort the population according to rank in ascending order of magnitude;
2. Assign a raw fitness to each solution by using a linear (not always the case) mapping function which interpolates from the lowest rank to highest rank;

3. Average the raw fitness of solutions with the same rank which ensure each of them have the same sampled-rate. Note that through the mapping and averaging procedure, as a result, the higher the rank is, the higher the fitness is assigned. Thus, those non-dominated solutions are emphasised in a population.

In order to maintain diversity within a population, Fonseca and Fleming (1993) introduced niche formation through a sharing function to solutions among each rank in their MOGA. This niche formation, introduced in order to maintain diversity, is called niche count. It provides an estimate of the extent of crowding near a solution (Deb, 2004). This fitness sharing procedure will be detailed in the next chapter for developing the MOGAST to solve multi-objective scheduling problem in capital goods industry.

The advantage of MOGA, as Deb (2004) stated is that MOGA can be easily applied to combinatorial optimisation problems since niche formation is used in objective space. MOGA also has a simple fitness assignment scheme and is suitable for problems which seek a spread of Pareto-optimal solutions.

However, all solutions, in a particular non-dominated front (except the first one) need not have the same assigned fitness, which observed by Deb (2004), may cause bias towards some solutions in the search space. The shared fitness procedure in MOGA does not guarantee a solution in a poorer rank always has a worse scaled fitness than every solution in a better rank (for example if a better rank has many crowded solutions it will lead to a very big niche count, then the scaled fitness of these solutions will be very poor) and therefore might lead to a slow convergence or inability to find a good spread in the Pareto front.

2.6.5 Non-dominated Sorting Genetic Algorithm (NSGA)

Most evolutionary multi-objective optimisation algorithms are required to find only the best non-dominated solutions in a population (Deb, 2004). These algorithms classify the population into two groups. They are non-dominated solutions and the remaining dominated solutions. MOGA is one of those algorithms. However, some algorithms require the entire population to be classified into various non-domination levels in an ascending order. Goldberg (1989) introduced this non-dominated sorting concept in his GAs. The best non-dominated solutions are classified as non-dominated solutions of level 1. Once the non-dominated solutions of level 1 are found, these solutions are

disregarded from the population. The non-dominated solutions of the remaining population are then found and thereby are classified as non-dominated solutions of level 2, and so on. The procedure continues until all individuals of a population are classified into a non-dominated level.

Inspired by Goldberg's (1989) idea of applying the non-dominated sorting concept in GAs, Srinivas and Deb (1994) proposed the Non-dominated Sorting Genetic Algorithm (NSGA). This NSGA uses a fitness assignment scheme that applies the non-dominated classification of solutions like MOGA. However, instead of just classifying solutions into non-dominated and dominated solutions, the NSGA classifies solutions into various non-dominated levels (shown in Figure 2-8). In order to preserve the diversity of the population, NSGA also applies a sharing strategy to each solution in the non-dominated front and carries a crowding distance assignment for calculating the density measure. If the solutions are classified into 4 fronts as shown in Figure 2-8, the solutions in front 1 are signed the highest fitness equally with respect to the importance of the closeness to the Pareto front. However, the 4 solutions unlikely represent or may say inadequately represent all the solutions in front 1. These solutions are located differently and biased with solution 1 represents nearly the whole top-left area, whilst solution 2, 3, 4 are squeezed in the bottom right. Solution 1 could easily disappear in the next generation after performing GA operator. As solution 1 represents a large area of the Pareto front, in order to maintain the diversity within the population and make sure solution 1 survives into the next generation, an extra fitness is needed to be added onto solution 1, so solution 1 is emphasized adequately. Hence a sharing function method was introduced for this purpose. This function relates to the crowding distance between solutions in the same front.

The main advantage of an NSGA is that the fitness assignment guarantees a good spread of solutions in a Pareto front that could be found by emphasising better non-dominated sets systematically and then maintains diversity among solutions. However NSGA is very sensitive to the parameter σ_{share} which is required to be fixed in advance (Srinivas and Deb, 1994), hence problems may arise when the niche size is to be determined.

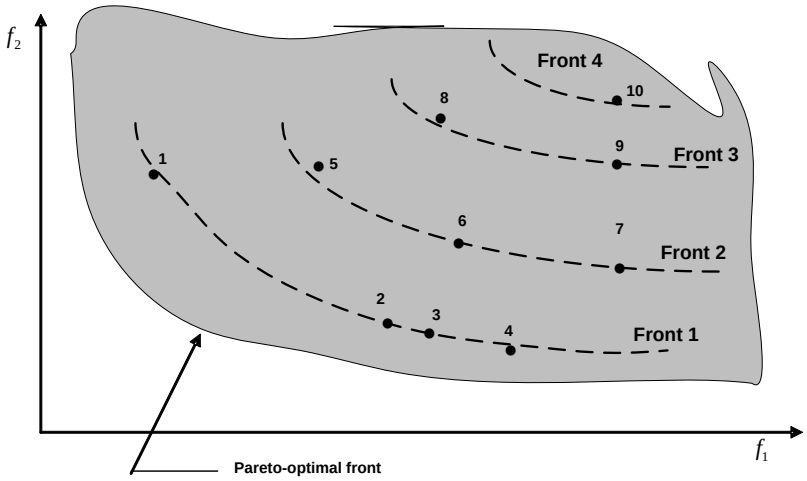

Figure 2-8: The classified non-dominated fronts(Deb, 2004)

2.6.6 Applications of MOEAs in production scheduling

Multi Objective Evolutionary Algorithms are particularly suitable for solving multi-criteria production scheduling problems and have been widely used. For example, Li et al. (2000) designed an extended model of MOGA approach to address earliness/tardiness production scheduling planning (ETPSP) with multi-process capacity balance, multi-product production and lot-size consideration. This MOGA was compared with Simple Genetic Algorithm (SGA). Their simulation results showed the MOGA is more effective and achieved better results. Ponnambalam et al. (2001) proposed a MOGA for job shop scheduling which considered three performance criteria: minimisation of the total makespan, total idle time of machines and total tardiness. They applied this MOGA on twenty-eight benchmark problems and the proposed MOGA was capable of providing optimal or near-to-optimal solutions. Pasupathy et al. (2006) proposed a MOGA for permutation flow shop scheduling which aimed to minimise the makespan and the total flow time of jobs. This MOGA was applied and evaluated in benchmark flow shop scheduling problems and yielded Pareto-optimal solutions. Yu et al. (2006) presented a MOGA for scheduling of a mix-model assembly line to level the part usage rate and minimise the makespan, and solving the multi-product dispatching problem of assembly systems. The computational results showed this MOGA presented satisfactory Pareto solutions. Tseng et al. (2008) proposed a hybrid evolutionary multi-objective algorithms (HEMOGs) for integrating assembly sequence planning and assembly line balancing. The HEMOGs searched out Pareto-optimal solutions effectively and contributed to references for the flexible change of assembly system design. Akgunduz and Tunal (2010) proposed an adaptive genetic algorithm approach to solve mixed-model assembly line sequencing problem where multiple objectives such

as total utility work, variation in part consumption rates and setup costs are considered simultaneously. The results showed that the proposed approach outperformed the non-adaptive algorithm in both solution quality and quantity. Jiang et al. (2011) proposed a multi-objective optimisation model that used a multi-objective genetic algorithm. The aim of this model was driven by avoiding inventory shortage whilst to meet requirements of production configuration changes. The impact costs, lead-time and inventory factors were considered in the new model compared to the normal configuration optimisation model. The practicality and effectiveness of this model was demonstrated in an industrial case study. However, the literature has not addressed the application of Multiple Objective Genetic Algorithms to the type of production scheduling problems encountered by capital goods companies.

2.7 A Brief Review of Artificial Immune System and Its Application on Production Scheduling

Artificial Immune System inspired algorithms (AISAs) are relatively new algorithms compared to GAs and have attracted ever increasing interest over the last fifteen years. The development of the AISAs has grown along with better understanding of Immunology. Unlike genetic algorithms, in which certain evolutionary elements work within a general framework, there exists no single algorithm from which all immune algorithms are derived (Greensmith et al., 2010). Hence there are many AISAs that have emerged which have been based upon theoretical immunology foundations. In immunology, it is believed that the human immune system is classified into two interactive sub-systems: the innate and adaptive immune system. The adaptive was first identified and viewed as more sophisticated and important than the innate system until recently, therefore theories such as clonal selection (Burnet, 1959) and negative selection (Lederberg, 1959) were first introduced. As the immunology developed, it was found that there are many issues that the adaptive immune system could not explain properly. Much of attention has been drawn back to the study of the innate immune system. The danger theory hence was proposed by Matzinger (1994) and danger theory formed the basis of algorithms such as Dendritic Cell Algorithm (DCA) (Greensmith and Aickelin, 2007). In addition, theories that attempted to explain the various emergent properties of immune systems have also emerged, such as the immune network theory (Jerne, 1974). Inspired by these theories, many algorithms have been proposed. The processes of the most popular algorithms for each theory are outlined in Appendix C.

The clonal selection theory inspired algorithms were popularly used in the literature. The clonal selection theory has two elements: clonal selection; somatic hypermutation and affinity maturation. By representing these two elements, clonal selection-based algorithms (CSAs) are constructed with a repeated cycle of match, clone, mutate, eliminate and replace. Firstly, B-cells match the antigen by binding the B-cell receptor i.e. antibody and a portion of the antigen called epitope. If the match occurs, the cell clones itself and the number of clone depends on its affinity. In order to bind the antibody more successful to antigen, hypermutation now plays its role and ensures that exact clones are not formed. This biological evolutionary element makes the affinity maturation process to generate the most responsive antibodies. Finally, the less responsive antibodies are eliminated and are replaced by newly generated ones. Then the whole process is repeated until the stopping criterion is met. A typical schematic of Clonal Selection Algorithm is shown in Figure 2-9.

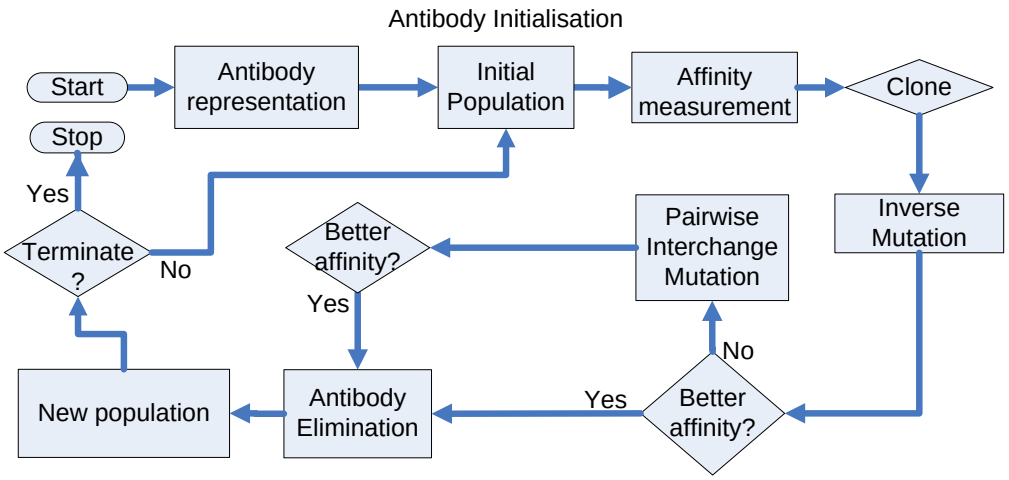

Figure 2-9. A schematic of Clonal Selection Algorithm

The CSAs have significant similarities to GAs. They both are population-based algorithms. They both apply affinity or fitness measurement to decide whether an immunological or evolutionary process applies or not on an antibody or chromosome. They both have biological evolutionary elements such as mutation and selection process. However, they have significant differentiations: first of all, GAs have crossover operators while CSAs do not. Secondly, while GAs usually have a static or fixed mutation rate, CSAs have a dynamic mutation rate and it generally decreases generation by generation.

Hart and Timmis (2008) raised the question as to whether AISAs added value to the

field of optimisation and stated in their work that evidence in the literature remained conflicting on various benchmark scheduling problems (Coello et al., 2003, Ong et al., 2005). Although AISAs were able to solve some problems better than others, they did not distinguish themselves as being significantly better than well established search techniques such as EAs. Nonetheless their arguments suggested that more theoretical study and applied research relating to the use of AIS for real-world problems should be carried out to gain more understanding. The new trends in this field include the adoption of hybrid approaches including approaches such as Evolutionary Algorithms, Simulated Annealing, Particle Swam etc (Ge et al., 2007, Kumar et al., 2006, Tavakkoli-Moghaddam et al., 2007, Zhang and Wu, 2008, Zhang and Wu, 2010).

AISAs have been successfully applied in production scheduling problems (Coello et al., 2004, Engin and Döyen, 2004, Ge et al., 2005, Xu et al., 2005, Chandrasekaran et al., 2006, Kumar et al., 2006, Kahraman et al., 2009, Zandieh and Gholami, 2009). Most of the literature focuses on flow-shop and job-shop scheduling. There were a few applications of AISAs in assembly planning (Cao and Xiao, 2007, Chang et al., 2009). There is lack of research in literature on applying AISAs for production scheduling of manufacturing environments that encountered in capital goods industry.

2.8 Identification of the Research Gap

Although the research gap was described and identified throughout the introduction of chapter 1 and the literature review in this chapter, it is worthwhile to summarise in this section.

With regard to production scheduling with a single objective, most of the research in the literature has focused on flow-shop and job-shop problems, and with only a few researchers who have studied assembly shops. There is a lack of research that has worked on production scheduling problems in the capital goods industry which involves component manufacturing and assembly together with a deep and complex product structure. Pongcharoen (2004) developed a GAST to solve these problems, however, this GAST was only able to solve relatively very small problems due to the limitation of its speed. The speed of GAST also limited the validation and analysis of the experiments as it include limited replications of each run. There was no practical tool that was able to solve extremely large scheduling problems in capital goods industrial within a realistic computation time. The GAST only used the Roulette Wheel as a

unbiased selection mechanism, biased selection mechanisms such as Elitism were not considered, which may enhance the performance of GA. Optimum Genetic operators were only determined through experiments that evaluated these operators separately with none or very limited number of replication. Combination of crossover, mutation and selection schemes may be determined simultaneously with experiments based on full factorial design. Artificial Immune System (AIS) has been widely studied and applied in production scheduling during the past fifteen years, however most of the research still focused on flow-shop and job-shop problems. There was no research observed in the literature that applied AIS to solve production scheduling in capital goods industrials. In the literature, suggestions were made that more work should be conducted to study and compare the performance of AIS and GA on various problems for a better understanding between these metaheuristic algorithms.

With regard to production scheduling with multiple objectives, most of the research in literature again focused on flow-shop and job-shop problems. There was no research that solved production scheduling problems with multiple objectives in capital goods industrials. The objectives of the research work in production scheduling literature were most likely the minimisation of total flow time, or makespan, or total tardiness of all jobs. Optimising delivery performance, inventory costs and resource utilization simultaneously were neglected in the literature which is crucial for the survival of capital goods companies in a global market. There was also lack of research in literature that evaluated the conflicting relationships among all these three objectives to justify the necessity of providing such tool for production scheduling. Research was also lacking in the production scheduling literature relating to how to determine the optimum configurations of the algorithm in terms of parameters such as niche size.

The above section described indicates that there is significant research gap in the literature for complex production scheduling in the capital goods industry, hence this was the main motivation of this work.

Chapter 3. Development of an Enhanced Single-Objective Genetic Algorithm Scheduling Tool

This chapter describes the development of an Enhanced Single-objective Genetic Algorithm Scheduling Tool (ESOGAST) for solving very large production scheduling problems in capital goods industry with multiple resource constraints and deep product structure. The ESOGAST consists of the following elements: input/output, graphical user interface, genetic representation, population initialization, genetic operations, repair processes, fitness measurement, selection scheme, parameter settings and stopping criteria. These elements are described individually. A detailed specification of the tool is also provided.

3.1 Overview

In this work, an Enhanced Single-Objective Genetic Algorithm Scheduling Tool (ESOGAST) was developed based upon the Genetic Algorithm based Scheduling Tool (GAST) developed by Pongcharoen (2001). The GAST was developed as an extension to a scheduling tool called the Gantt program developed by Hines (1996) and was written in the Tcl/Tk programming language (Ousterhout, 1994). The GAST provided a Genetic Algorithm for solving the scheduling problems encountered by capital goods companies and was designed to show a schedule as a Gantt chart. It converted the planning information from a data file generated by the Manufacturing System Simulation Model developed by Hicks (1998).

The ESOGAST developed in this work was designed to minimise tardiness and earliness penalties for early or late arrival of components and parts and final products. The ESOGAST maintains most elements of GAST with the same fitness function, but has an enhanced repair process that increased computational efficiency in order to solve very large problems. It also has many more additional selection schemes (see section 3.2.7), such as Tournament Selection(Goldberg, 1989), Elitism Strategy (De Jong, 1975) etc. The ESOGAST was written in C (Kernighan and Ritchie, 1988), which is a compiled language that runs much more quickly than an interpreted language such as Tcl/Tk. The speed of ESOGAST enabled it to solve every large problem that was available from data obtained from the collaborating company. It was also possible to conduct comprehensive experiments with a full factorial design (Montgomery, 1997) These experiments investigated many levels of parameter settings with sufficient

replications to allow robust statistical analysis. The new program aimed to produce optimal schedules using data files generated by the Manufacturing System Simulation Model developed by Hicks (1998).

3.2 Elements Development of an Enhanced Single-Objective Genetic Algorithm Scheduling Tool (ESOGAST)

This ESOGAST algorithm includes the following GA elements: gene encoding, genetic representation, population initialization, genetic operations, repair processes, fitness assignment and measurement, selection scheme and stopping criteria. The algorithm is illustrated in Figure 3-1.

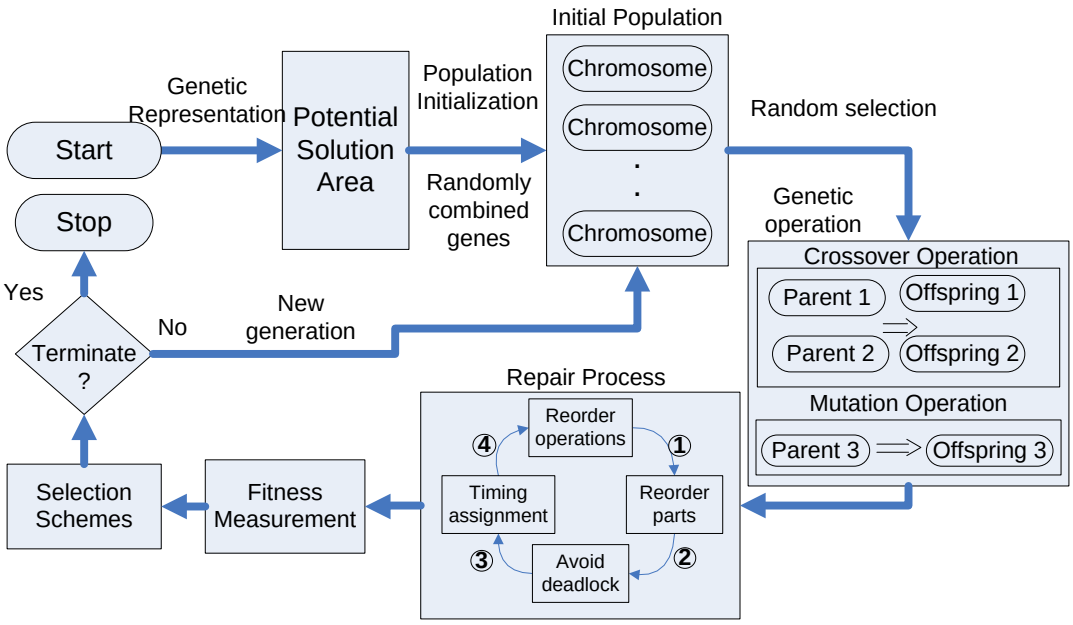

Figure 3-1. Structure of ESOGAST for production scheduling

3.2.1 Encoding scheme of product structure

Hicks (1998) developed a simulation system of ETO/EMO manufacturing for capital goods companies. The product structure identifier (PSI) and product instance identifier (PII) proposed by Hicks (1998) for his work were applied in this work. The part number refers to a particular type of component or assembly. The product structure and instant identifiers and operation number are used to retrieve the data. The Encoding scheme of product structure is illustrated in Figure 3-2 as the node numbers and the root node represents the final product. The PSI represents the relations between components and assemblies in different levels of product structure. For example, PSI=1:2:5 represents the relationship between part number 1, 2 and 5. Part number 1 is the parent part of part

number 2 and part number 2 is the parent part of part number 5. In reverse, part number 5 is the child part of number 2 and part number 2 is the child part of part number 1. The product instance identifier identifies and distinguishes between different instances of identical parts, as sometimes an assembly may contain more than one item of the same type. For example, the product with part code 1 contains two identical assemblies with part code 2 and each of the part code 2 has 2 parts: part code 5 and 6. For part 5 and part 6 in each part code 2, the PSIs are same here. However the PIIs are different. Applying PSI and PII, this coding scheme uniquely identifies the location of each type of part/assembly within the product structure and also facilitates the data processing in the program.

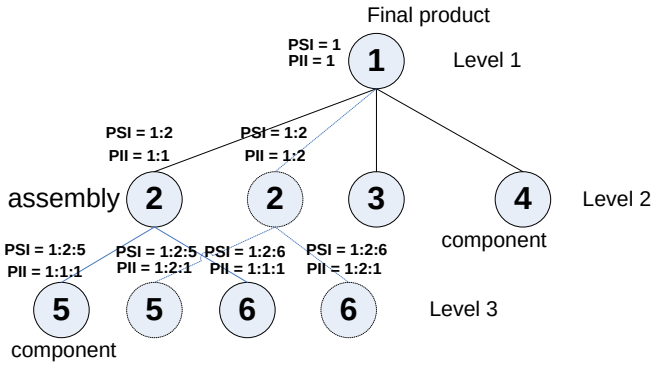

Figure 3-2. Encoding scheme for the product structure (Pongcharoen et al., 2004)

3.2.2 Gene encoding

The sequence of operations within each production schedule is encoded as a string. The representation of a gene used by Pongcharoen is adapted and is presented in Figure 3-3. The gene contains the product structure identifier number, the product instance identifier number and the operation number, which can be used to search for other detailed information. Such information includes part description, transfer time and operation time etc.

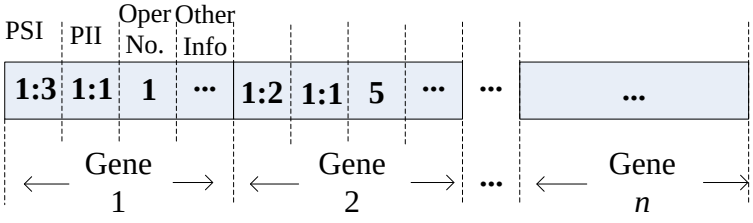

Figure 3-3. Gene representation (Pongcharoen et al., 2004)

3.2.3 Genetic representation and population initialisation

A chromosome representation is shown in Figure 3-4. Each chromosome consists of *n* sub-chromosomes which represents *n* resources (machines). Each sub-chromosome is combined with *m* genes (*m* may be different in each sub-chromosome and the number of genes or length of sub-chromosome depends on the size of the problem). A whole chromosome then represents the complete sequence of operations within the production schedule. Each sub-chromosome is generated by randomly selecting genes. This process is repeated until population size has been reached. Each machine has different capabilities that a particular group of operations of various parts are only able to be performed on a particular machine. Hence genetic operations are performed within the same sub-chromosomes and genes cannot be swap among different sub-chromosomes.

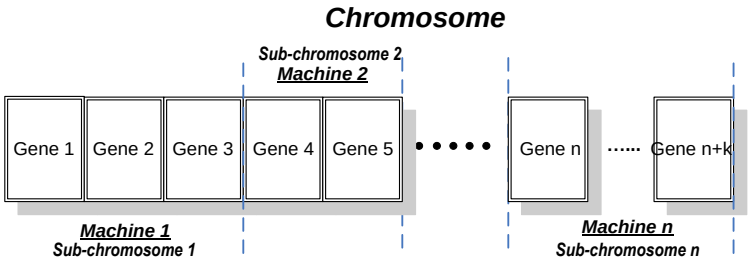

Figure 3-4. Chromosome representation (Pongcharoen et al., 2004)

Note that the overall population structure is built up within a double linked chromosome structure that is illustrated in Figure 3-5. The whole structure represents an entire population, with each row representing a particular chromosome (schedule. Within this structure, every gene is double linked using pointers and every gene is easily located within a population by the pointers. The use of pointers means that the data structures are stored with the computer's memory, which helps minimise search times. There is an initial chromosome created which is separated from chromosomes within a population. Each gene within this chromosome contains all the necessary information which is retrieved from data files.

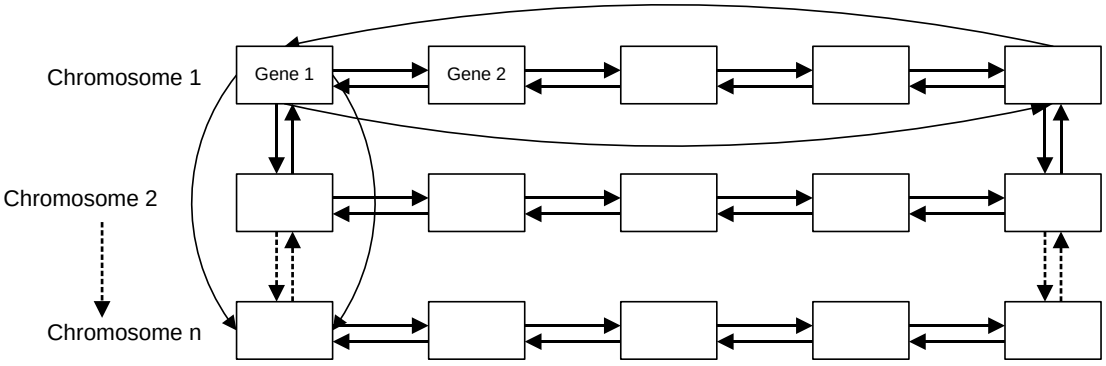

Figure 3-5. The representation of population structure

3.2.4 Genetic operation

After the population is initialised, a specified percentage of chromosomes are selected to be subject to crossover and mutation genetic operators. The probability of crossover (%C) and mutation (%M) are important experimentation parameters. For crossover two parent chromosomes are randomly selected, whereas mutation only requires one chromosome. The crossover and mutation operators used in this research are illustrated in Table 3-1 and Table 3-2.

Initial	Description (Crossover)
1PX	One Point Crossover (Murata and Ishibuchi, 1994)
AEX	Alternating Edges Crossover (Grefenstette, 1986)
CYX	Cycling Crossover (Oliver et al., 1987)
ERX	Edge Recombination Crossover (Whitley et al., 1989)
MPX	Maximal Preservation Crossover (Muhlenbein et al., 1988)
OX	Order Crossover (Davis, 1985)
PBX	Position Base Crossover (Syswerda, 1991)
PMX	Partially Mapped Crossover (Goldberg and Lingle, 1985)
2PCX	Two Point Centre Crossover (Murata and Ishibuchi, 1994)

Table 3-1 Crossover operators

Initial	Description (Mutation)
CIM	Centre Inverse Mutation (Tralle, 2000)
2OAS	Two Operations Adjacent Swap (Murata and Ishibuchi, 1994)
3OAS	Three Operations Adjacent Swap (Murata and Ishibuchi, 1994)
2ORS	Two Operations Random Swap (Murata and Ishibuchi, 1994)
3ORS	Three Operations Random Swap (Murata and Ishibuchi, 1994)
IM	Inverse Mutation (Goldberg, 1989)
DM	Displacement Mutation (Michalewicz, 1992)
SM	Scramble Mutation (Syswerda, 1991)

Table 3-2 Mutation operators

3.2.5 Repair Process

The crossover and mutation operators may produce infeasible schedules that contravene operation or assembly precedence relationships or that are not possible due to finite capacity constraints of machines. Infeasible chromosomes are rectified using a modified

version of the four stage repair process developed by Pongcharoen (2001) which was developed to enable the algorithm to tackle more complex production scheduling problems. The modified repair process developed in this work is described as follows (note that the repair process also works within sub-chromosomes):

1st stage: Operation precedence adjustment.

All the operations must be performed in a logical order which in this research is assumed to be sequential order i.e. the second operation can only be performed after the first operation is finished. The third operation can only be performed after the second operation is finished and so on. The sequences of operations of a particular part are checked across machines because the operations of a part may be performed at different machines. If the GA produces schedules where the operations are not in the correct order, an adjustment process is carried out to reorder the sequences.

This procedure starts by checking the number of operations required for each part. If the number is more than one, the sequences of operations need to be checked. If the operations are not in a sequential order, they are replaced by those stored in the right order in the initial chromosome. Those operations in a right order remain unchanged.

2nd stage: Part precedence adjustment.

The supply of component parts and subassemblies must be co-ordinated with subsequent assembly processes. For example, the parent assembly cannot not be completed until all its subassemblies are completed. The algorithm corrects such problems by adopting a 'bottom up' strategy in which items at the bottom of the product structure are scheduled first, the algorithm then moves up the product structure level by level. Note that product level depends on levels of assembly and parts, not on operations.

The product level of each part is stored as gene information. This procedure starts at the first gene of each sub-chromosome and checks the product level and compares it to the product level of the successive ones. If the product level of previous gene is higher than the successive one, the order remains the same; otherwise the genes will be swapped. The procedure then starts with the second gene and the process is repeated until the last one. After this process, all the operations associated with a higher level of product structure are placed in front of the operations associated with a lower level of product structure. Note that this procedure does not apply for operations when their final

products (root number) are not the same, hence the product structure level of these parts are not relevant. This is ensured by checking the root number of each gene.

3rd stage: Deadlock adjustment

When scheduling multiple machines it is possible that two machines may become deadlocked because they both are waiting for a part from each other. Deadlock adjustment identifies deadlock situations and starts from the first gene of all sub-chromosomes by checking the precedence relationships between operations. Operations without precedence relationships can be performed and are marked and added to a list of legal operations. Again, starting from the first sub-chromosome, the algorithm then moves to the first unmarked gene within the sub-chromosome to check whether its previous operation appears in the list of legal operations. If so, it is added to the list. Otherwise, the procedure moves to the next sub-chromosome and the rest and repeats the above procedures to find a legal operation. If there is no operation that can be added to the list of legal operations, a deadlock situation has been identified. Then a sub-chromosome will be randomly selected to find the operation that can be added on the list. This procedure will go on for every sub-chromosome until a legal operation is found. This operation is then moved into the position of the first deadlock operation (unmarked operation) within this sub-chromosome and then added on the list of legal operations. The order of operations after this deadlocked position will remain the same. The process is then repeated until all operations appear in the list of legal operations. If a deadlock is not found, the schedule is considered feasible; hence no deadlock adjustment is required.

4^{th} stage: Timing assignment and capacity considerations

The above procedures only produce chromosomes that represent sequences of operations. It is necessary to assign timing information to produce a schedule. After the previous processes are performed, the sequences of operations which are represented by the genes in each chromosome can be assigned timing sequentially without any deadlock. In scheduling, schedules aim to minimise the idle time between operations, i.e. perform an operation straight after the previous one as soon as possible. However, the finite capacity constraints may cause a delay between operations. The timing assignments are determined by the duration of operations including set-up, machining and transfer time.

3.2.6 Fitness assignment and measurement

After the repair process, the next step is to perform the fitness assignment procedure. The fitness function proposed by Pongcharoen (2001) is adopted. It is shown in Equation 3-1. Note that the lateness and earliness penalty rate set for this research was £100 and £50 per day. The value was based upon previous work conducted by Pongcharoen (2001). The penalty rates used in this research indicates the rate used by Pongcharoen was likely too high based on the high value of penalty obtained for the full size problem. However, the penalty ratio between tardiness and earliness were unchanged as 2:1. This will make sure the schedules produced by ESOGAST will not be affected when compared to those produced by GAST (Pongcharoen, 2001).

$$Total\ cost = tardiness\ penalties + earliness\ penalties$$

$$F = P_t \sum_{i=1}^{n} T_{pi} + P_e\left(\sum_{i=1}^{n} E_{ci} + \sum_{i=1}^{n} E_{ai} + \sum_{i=1}^{n} E_{pi}\right) \qquad \text{Equation 3-1)}$$

Where $T_{pi} = \max(0, F_{pi} - D_{pi})$ $E_{pi} = \max(0, D_{pi} - F_{pi})$

$E_{ci} = \max(0, D_{ci} - F_{ci})$ $E_{ai} = \max(0, D_{ai} - F_{ai})$

Notation: subscript *a, c, p, i* (assembly, component, product, item number), *n* is the number of the according items (either assembly, or component, or product)

$P_t = Penalty\ rate\ of\ tardiness\ (pounds\ perday)$

$P_e = Penalty\ rate\ of\ earliness\ (pounds\ per\ day)$

$T_{pi} = Tardiness\ of\ product\ i\ (days)$ $E_{pi} = Earliness\ of\ product\ i\ (days)$

$E_{ci} = Earliness\ of\ component\ i\ (days)$ $E_{ai} = Earliness\ of\ assembly\ i\ (days)$

$F_{pi} = Finish\ time\ of\ product\ i\ (date)$ $D_{pi} = Due\ date\ of\ product\ i\ (date)$

$F_{ci} = Finish\ time\ of\ component\ i\ (date)$ $D_{ci} = Due\ date\ of\ component\ i\ (date)$

$F_{ai} = Finish\ time\ of\ assembly\ i\ (date)$ $D_{ai} = Due\ date\ of\ assembly\ i\ (date)$

3.2.7 Genetic selection scheme

In this research, the selection schemes shown in Table 3-3 are used to choose the chromosomes that survive to the next generation. The Roulette Wheel approaches use a random number generator over the range 0-1. The probability of an individual surviving to the next generation is determined by its fitness. In Tournament selection, two randomly selected chromosomes compete for succession to the next generation. The elitist strategy selects the best chromosomes from previous generation to the next generation. However the percentage selected is an experimental parameter. The roulette-

elitist and rank-based roulette-elitist strategies combine the elitist strategy and roulette wheel approaches. The first one uses a standard roulette wheel approach, the latter uses rank-based roulette wheel approach. In the Rank-based Roulette-Elitist strategy (Teerawut, 2008), it was found that 15% of the best individuals in the current generation selected by the Elitist Scheme for the next generation produced the best results. It was therefore chosen for experiments in this research.

Initial	Description
SRW	Standard Roulette Wheel (Goldberg, 1989)
RRW	Rank-based Roulette Wheel (Reeves, 1995)
TS	Tournament selection (Goldberg, 1989)
SRSR	Stochastic Remainder Sampling without Replacement (Goldberg, 1989)
ES	Elitist Strategy (Goldberg, 1989)
RES	Roulette-Elitist Strategy (Teerawut, 2008)
RRES	Rank-based Roulette-Elitist Strategy (Teerawut, 2008)

Table 3-3 Selection schemes

3.3 Specification of the ESOGAST Model

This section provides a functional specification of ESOGAST and summarises its construction and technical features. This tool aims to solve very complex production scheduling problems in capital industry and it is optimised for speed. A flow chart of the ESOGAST is also presented.

3.3.1 General principles

A number of general principles have been applied in the design and construction of the ESOGAST model. Existing data is extracted from the data files created by Hicks (1998) and then transferred into text files. These data was edited to text files with a ready-to-loaded format. The ESOGAST program then loads these files for program execution. The program has been written and edited using Dev-C++. The program can be run on using Dev-C++ on a computer with either a Linux or Microsoft operating system. The specification of the PC used for the work is: AMD Athlon(tm) Dual Core Processor 4450B 2.30GHZ with memory RAM of 2.00GB. Operation system: English Windows Vista Enterprise 32-bit Service Pack 2 (which is quite a modest machine by modern standards). The specification of the time sharing machine running Linux used for the work is: dual 3GHz Intel CPU servers with 12 GB of memory. When the ESOGAST

was being executed, the user is asked to upload the name of the file. The program then will automatically calculate the chromosome length, number of machines, total number of parts and assemblies etc. Then a series of parameter settings will be requested: population size, number of generations to be run, genetic operators to be chosen from the list, penalty rate etc.

3.3.2 Data Structures

The data that is used by the ESOGAST is obtained from a real world manufacturing systems within a collaborating company. The original data was stored as binary code in data files by Hicks (1998) which were produced using the Pascal language (Wirth, 1971). All the data used for this research was retrieved again using C language (Kernighan and Ritchie, 1988). The data includes static data and operational data. During the scheduling process, the static data may not be changed. This includes product structure (the parent and children of the item, the number of the children of each product), product number, name, and description, instant identifiers, part routing (resource code and operation sequence). The operational data, that may be changed during the scheduling process, includes the schedule start time, set up, machining and transfer time of all operations, and due date etc.

3.3.3 Input/Output files

The ESOGAST input data file may be retrieved from the Manufacturing System Simulation Model developed by Hicks (1998). The data then is edited and formatted into the following fields in text files, ready to be uploaded by the ESOGAST program:

1. Part number;
2. Part instance number;
3. Operation number;
4. Part description
5. Part code;
6. Number of operations for each machine;
7. Set up time;
8. Machining time;
9. Number of transfer;
10. Transfer time;
11. Product structure level;
12. Machine number;

13. Part root number;
14. Parent part number;
15. Parent instance number;
16. Due date;
17. Product family structure number;
18. Start time;
19. Auditing period.

The program produces two output files when execution is finished. One is a text file that was formatted into fields necessary for upload into Pongcharoen's Gantt chart program (2001) to product a Gantt chart. These fields are:

1. The starting time;
2. The bucket time or auditing period;
3. Part ID;
4. Part code;
5. Part description;
6. Part family;
7. Parent part;
8. Number of operations on routing (routing length);
9. Resource number;
10. Starting time for set-up;
11. Set-up time;
12. Starting time for machining;
13. Machining time;
14. The number of transfers;
15. Starting time for transfer;
16. Transfer time;
17. Part due time;
18. Original due time.

Another output file that produced by ESOGAST program is a summary text file. The file is able to show the following information and used for experimental analysis:

1. The file uploaded in this run indicating problem type and size;
2. The configuration of the ESOGAST applied in this run (GA configurations and penalty rate etc.);

3. The minimum, mean and maximum penalty and standard deviation of each population;
4. The best chromosome found, the generation that it is found and its total penalty cost;
5. The execution time of each run.

3.3.4 Validation of the ESOGAST

Model validation is defined as 'substantiation that a computerised model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model' (Schlesinger, 1980). A validated model gives users a willing to use it with confidence. Schlesinger's (1980) triangle validation model was considered in this research. This model covers three stages of the modelling process (see Figure 3-6).

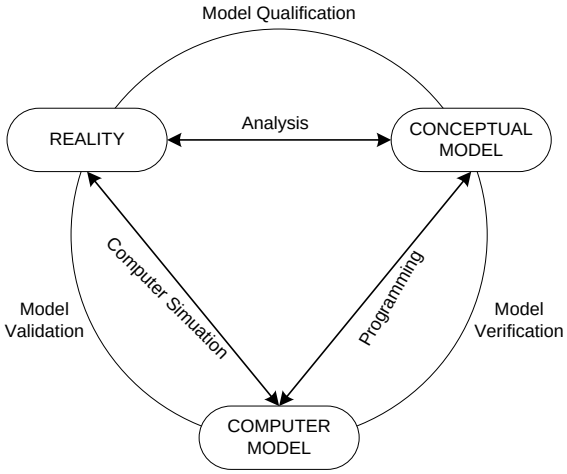

Figure 3-6 The validation of simulation models (Schlesinger, 1980)

1. Model qualification determines the adequacy of the conceptual model (Hicks and Earl, 2001). The validity of the conceptual model also determines the theories and assumptions of the model are correct and that the representation of the problem reality and the mode's structure, logic, and mathematical and relationships are reasonable and acceptable for the use of the model (Robert, 1983). In this research, the conceptual model contains elements such as product representation and resource description. The product representation represents a hierarchical description of the product structure and relationships among different products (Pongcharoen, 2001). The resources also contain a hierarchical structure which is viewed as a hierarchy of levels of organizations. This enable each resource is easy to be identified (Hicks and Braiden, 2000).

The ESOGAST program is a further development of Pongcharoen's GAST (2001) and both adopt the same conceptual model.

2. Model verification is a process that ensures the computer program functions correctly within specified limits of accuracy using carefully chosen test cases. Although it is not practically possible to prove that the validation of the code due to the complexity of the code and the schedules it produced, it is possible that by conducting large number of tests on relatively small problems, each of which can prove the program is invalid (Pongcharoen, 2001). These extensive tests give confidence of the program and the output file hence may be considered reliable. During the model verification process, a large number of tests are conducted and compared with small hand simulations examples to ensure it produces legal schedules. The output files also can be tested by uploaded by GAST. Output files are repeatedly overall checked and performance measure is correctly calculated.
3. Model validation determines that the model's input/output behaviour has the accuracy required for the model's purpose. After the program satisfies the previous two processes, model validation tests the input/output transformation of the program by statistical analysis (Hicks and Earl, 2001).

3.3.5 Functional specification of the ESOGAST program

The sequences of the ESOGAST program procedure are shown as follows:

1) Parameter setting: the program enables the user to input the parameters as desired. These parameters are: problem size, population size, number of generation run, genetic selection scheme, crossover operator, crossover probability, mutation operator, mutation probability; (go to 2);
2) Population initialisation: there are two steps to initialise population.
    a) A string of genes (chromosome) is generated and stored, which contains the data information from the input file. This single chromosome is the initial chromosome to create a population and it uses double-linked structure. (go to 2b);
    b) All the genes within its sub-chromosome of this initial chromosome are non-repeatedly and randomly selected and copied one by one. These genes then are put together to generate a new sub-chromosome. After all the new sub-chromosomes are generated, a new chromosome then is formed. (go to 2c);

c) Step 2b is repeated until the number of chromosomes generated equal to the specified population size. The initial population is then produced. (go to 3);

3) Genetic operations: chromosomes are randomly selected from the initial population to perform genetic operations as follows.
   a) The number of chromosomes perform crossover operation can be calculated by multiply the probability of crossover with population size. (go to 3b);
   b) Two chromosomes are randomly selected to perform specified crossover operation. (go to 3c);
   c) Repeat 3b until the number of children equals population size times crossover probability. (go to 3d);
   d) A chromosome is randomly selected to perform specified mutation operation. (go to 3e);
   e) Repeat 3d until the number of children equals population size times mutation probability. (go to 3f);
   f) Check if the population size is met or not. If not, a chromosome is randomly selected from the initial population and place to the new population. (go to 3g)
   g) Repeat 3f until the number of children meets the desire population size. (go to 3h) ;
   h) All the produced children are place in a temporary population. (go to 4);
4) Repair process: the repair process is introduced on this state in order to rectify the infeasible chromosomes produced in step 3) so the fitness evaluation can be performed next. The repair process is described as follows:
   a) Each chromosome is checked for the number of operations required for each part. If there is more than one operation, the sequence of operations needs to be checked. If the operations of a part are not in the sequence, they are replaced by a string of sequential operations of the part which stored in the initial chromosome. The procedure is repeated for all parts. (go to 4b);
   b) Step 4a is repeated for all chromosomes. (go to 4c);
   c) The procedure starts at the beginning of the first sub-chromosome and checks the level of each item in the product structure. This approach places operations associated with higher levels of product structure at the beginning of the sequence while operations associated with lower levels of product structure at the end of the sequence within the sub-chromosome. This procedure is repeated for all sub-chromosomes. (go to 4d);
   d) Step 4c is repeated for all chromosomes. (go to 4e);

e) The procedure starts at the first gene of the first sub-chromosome. The operation will be added onto the legal operation list and marked if it is without any precedence relationship. The logic then moves to the next gene, the above procedure is repeated until there is no legal operation is found within this sub-chromosome. The position of this gene is marked as deadlock position. Note that in order to be added on the legal list, the previous operation of a part in position has to be on the legal list already. Then the logic moves into the first gene of next sub-chromosome and repeats the procedure performed on the previous sub-chromosomes till all the sub-chromosomes are checked. After all the sub-chromosomes are checked which means a loop is finished, there are two situations could be encountered. One is all the operation can be added on the legal list, hence there is no deadlock (this will only likely happen if there are very few operations on the machines). Or the legal operation cannot be found to be added to the legal list, hence there is a deadlock and then the deadlock adjustment will be called. The deadlock adjustment will start from the deadlock position of a randomly selected sub-chromosome, search and identify the next legal operation by checking operations on the legal list. If there is a legal operation found in this sub-chromosome, this operation will also be added on the legal list. The legal operation found will swap the position with the gene in deadlock position within the sub-chromosome. (go to 4f);

f) Step 4e is repeated till all the operations are added onto the legal list. (go to 4g);

g) Timing assignment and finite resource capacity are considered in this step. In this case, the start set-up operation cannot take place until both the previous operation on the part and the previous operation on the same resource are finished. This timing assignment is performed and timing is assigned according to the sequence of the legal operation list. The process is repeated until all operations have been assigned. (go to 4h);

h) Step 4g is repeated for all chromosomes. (go to 5);

5) Fitness evaluation: three fitness functions are used to evaluate the penalty of each chromosome respectively in order to calculate the fitness of chromosomes. It has the following sequences:

a) The finish time of the final product is compared with its due date. If final product is finished late, lateness time is calculated. Earlier completion time for every component, assembly and final product is compared with their due date and is accumulated. The penalty cost associated with the schedule can be

determined by the lateness and earliness times as described on Equation 3-1 in section 3.2.6 (go to 5b);

b) Step 4a is repeated for all chromosomes. (go to 5c);

c) Find the best chromosome with least penalty cost of the whole population and stored. The minimum, mean and maximum penalties of the population are also calculated and stored. (go to 6);

6) Chromosome selection: based on the selection scheme chosen in step 2), the selection schemes are applied accordingly. The successive chromosome is selected and placed in the initial population. This procedure repeats until the desired population size is met and the previous initial population is replaced. (go to 7);
7) Program termination: termination condition is checked. If the condition is satisfied, go to step 8. Otherwise, next generation is required. (go to 3);
8) Displaying the result: two output file are produced. (go to 9);
9) The ESOGAST is terminated.

Figure 3-7 demonstrates the flow chart of the ESOGAST program.

3.3.6 Assumptions within the ESOGAST model

The assumptions used in Pongcharoen's (2001,p54) work are used for the ESOGAST model in this work. These assumptions were widely applied in the field and the static scheduling process of this kind also emphasises the importance of the speed of the tool, so quick adjustments or re-scheduling could be make in a short period of time if necessary in a dynamic scheduling environment.

1) Each operation cannot be performed until the completion of its predecessor providing the availability of a machine. This is known as a semi-active schedule (Conway, 1967, Baker, 1974).
2) Only one operation can be operated on a machine at a time. This is known as finite capacity (Jain and Meeran, 1999).
3) Each operation can be performed only on one machine at a time (Blazewicz et al., 1996).
4) There is no interruption of operation process such as machine breakdowns (King and Spackis, 1980).
5) There is no rework allowed (Ramasesh, 1990).

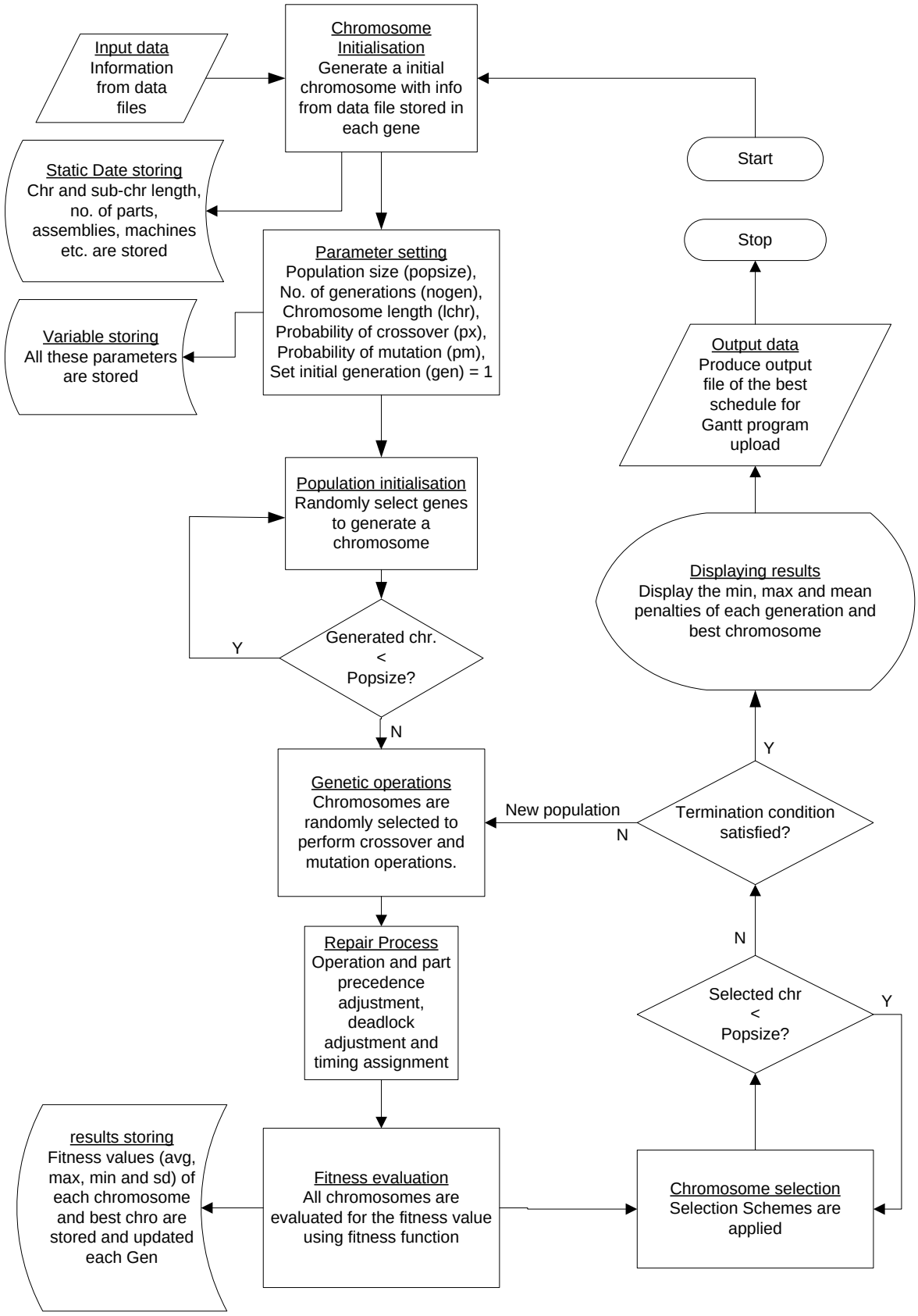

Figure 3-7: Flow chart of the ESOGAST program

Chapter 4. Experimental Programme I

This chapter describes the experimental programme that was designed to investigate the ESOGAST. The experimental programme had four objectives: 1) to compare the performance of ESOGAST with the GAST in terms of computational efficiency and the quality of the solutions produced; 2) to compare the performance of the selection schemes developed in this work with previous methods; 3) to compare the performance of different combinations of crossover and mutation operators; and 4) to solve a very large scheduling using the best configuration of Genetic Algorithm operators, selection schemes and parameters are base on results from previous experiments.

4.1 Experimental Design

The experimental programme was based upon five industrial problems summarised in Table 4-1. The small, medium, large and extra large problems are subset of the full size problem. This also gives confidence to the determination of parameters for the full size problem by studying these four problems. All the experiments were based upon a full factorial design using the factors and levels shown in Table 4-2. Compared to one factor at a time experiment design, Montgomery (1997) argued that full factorial experimental design is more efficient, and is necessary when interaction among factors exist and will allow the effects among factors to be estimated. In this research, this technique gives confidence to the investigation of determining optimum parameters of Genetic Algorithm without any pause or disruption.

Problem types	No. of Parts	Operations/assembly /products	No. of Machines Used	Levels(product structure
Small	15	25/9/2	8	4
Medium	18	57/10/2	7	4
Large	29	118/17/2	17	4
Extra large	85	268/39/1	25	7
Full size	1017	2442/278/57	36	8

Table 4-1 Characteristics of production scheduling problems

Factor	No. of levels	Specified levels uesed
Population size	3	60, 80, 200
Number of generations	2	20, 100
Probability of crossover	2	0.6, 0.8
Probability of mutation	2	0.08, 0.18
Crossover	9	See Table 3-1
Mutation	8	See Table 3-2
Selection scheme	7	See Table 3-3

Table 4-2 Experimental factors

4.2 A Comparison of ESOGAST with GAST

The results from ESOGAST were compared with those obtained by Pongcharoen's GAST (Pongcharoen, 2001). In this Experiment, three different problems (small, medium and large) were investigated. To compare the computational efficiency an experiment was conducted that considered the extra large problem with a population size of 60 and 20 generations and the same GA parameters on the same computer. The GA configuration used was the optimal configuration use by GAST. The extra large problem was chosen because the computation time of ESOGAST was relatively longer; hence it is more accurate (CPU time of ESOGAST for other problem were all less than one second). The execution time for GAST was 3936 seconds and for the ESOGAST was within 3~4 seconds (see Table 4-3), the ESOGAST was therefore 1,312 times more efficient. The fast processing speed of ESOGAST makes it possible for it to solve considerably larger problems within reasonable time. It also became feasible to significantly increase the amount of search by using larger populations and more generations.

Problem size: Extra large (Product 227)	
GAST	ESOGAST
Computational time = 3936s	Computational time = 3~4s
Times fastened: 3936/3~4 = 984~1312	

Table 4-3 Comparison of running time consumed

The next stage of this experiment compared the performance of GAST and ESOGAST in terms of the minimum penalty cost. The GAST was configured to use a population size of 60 with 20 generations. To take advantage of its increased efficiency the ESOGAST was configured to have a population size 200 with 100 generations. Optimum GA configurations were used by GAST and ESOGAST based upon experiments conducted by Pongcharoen (2001) and Xie et al (2011) respectively. Genetic operators ERX and 2OAS, with a probability of crossover of 0.6 and probability of mutation of 0.18 were used by GAST. Genetic operators PBX and 2OAS, with a probability of crossover of 0.9 and probability of mutation of 0.01 were used by ESOGAST. The ESOGAST also used the new RRES selection scheme, which had been found to be superior to the Roulette Wheel by experiment in Section 4.4. The results are shown in Table 4-4. Compared to the GAST, the ESOGAST achieved reductions in minimum penalty cost of 5.3%, 31% and 46% for the small, medium and large problems respectively.

	GAST			ESOGAST		
Problem size	Small	Medium	Large	Small	Medium	Large
Minimum Penalty Cost (£)	4216	4526	14521	3993	3125	7821
Improvement (%)				5.3	31	46

Table 4-4 Comparison of performances (GAST Vs. ESOGAST)

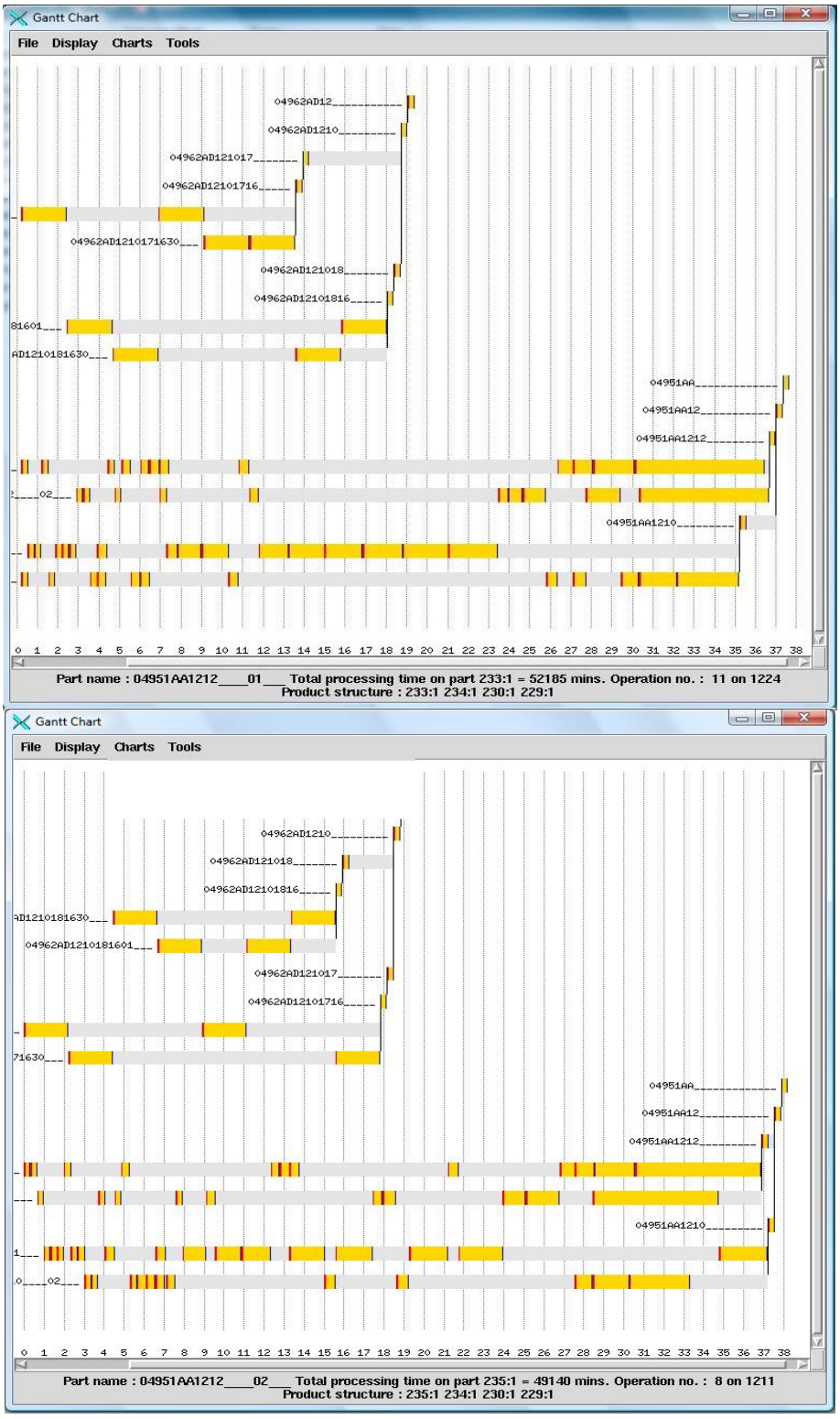

Figure 4-1. Gantt chart comparison (GAST Vs. ESOGAST)

Figure 4-1 shows two schedules shown as Gantt charts, with the top one produced by GAST and bottom one produced by ESOGAST. It can be observed that although two products were completed in the same time, the completion time of assemblies and components in the bottom Gantt chart contains less slack.

4.3 A Comparison of the Performance between Random Search with Repair Process and ESOGAST

Pongcharoen et al. (2004) identified the requirement for a repair process. Due to the complexity of production scheduling in capital goods industrial, the repair process is an essential aspect of the ESOGAST to rectify infeasible schedules. However, the repair process reorders the sequences of genes after each genetic operation; it may degrade the power of the genetic operators within the GA. In order to test the effectiveness of the GA it was compared with a random search followed by a repair process.

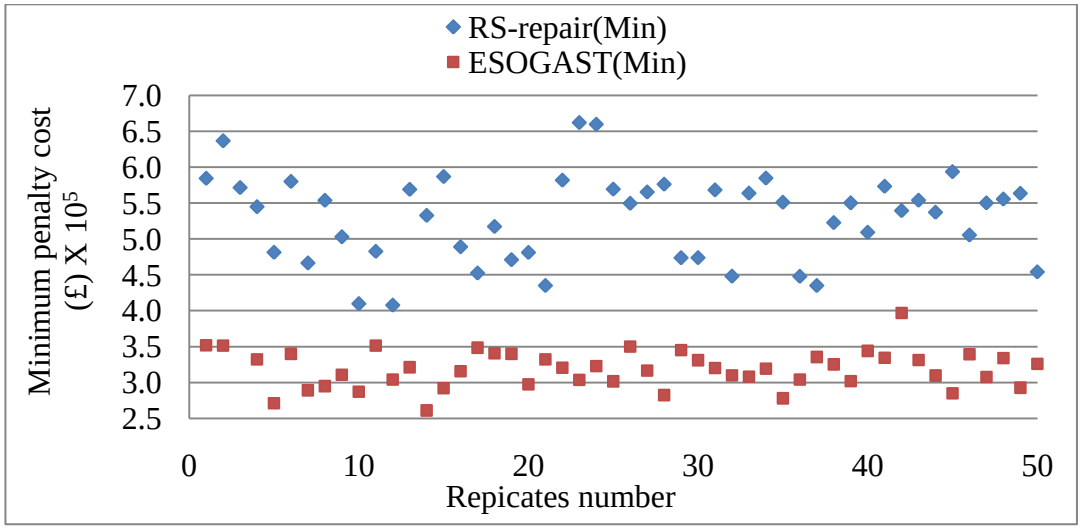

Figure 4-2. Comparison between random search and GA (both with repair process)

For this experiment the extra large problem was considered because it had the largest search space. Both the ESOGAST and random search with repair process used the same GA configuration. Fifty replications were considered. Figure 4-2 illustrates the performance between random search with repair process and GA with repair process. The ESOGAST produced superior results of minimum penalty costs, when compared to a random search with repair process. This demonstrates the advantage of using Genetic Algorithms and it also suggested that the repair process did not degrade the power of the GA because after performing repair process, GA still outperforms random search.

4.4 An Evaluation of Alternative Selection Schemes

In this experiment, the relative performance of the seven selection schemes shown in Table 3-3 was evaluated. This analysis was also based upon the extra large problem because of its large search space. A population size of 80 was used together with 20 generations. The crossover and mutation operators were 1PX and CIM. Crossover and mutation probabilities were 0.8 and 0.08. These particular values were based upon the results of Pongcharoen (2001). Each run with a specified configuration was replicated 50 times with different seed numbers. Figure 4-3 shows the mean of minimum and mean penalty costs achieved by the various selection schemes. It can be seen the Elitist Strategy with Rank-based Roulette Wheel achieved the lowest value both in terms of mean and minimum penalty. It also can be seen that all biased selection schemes achieved lower penalties compared to Standard Roulette Wheel.

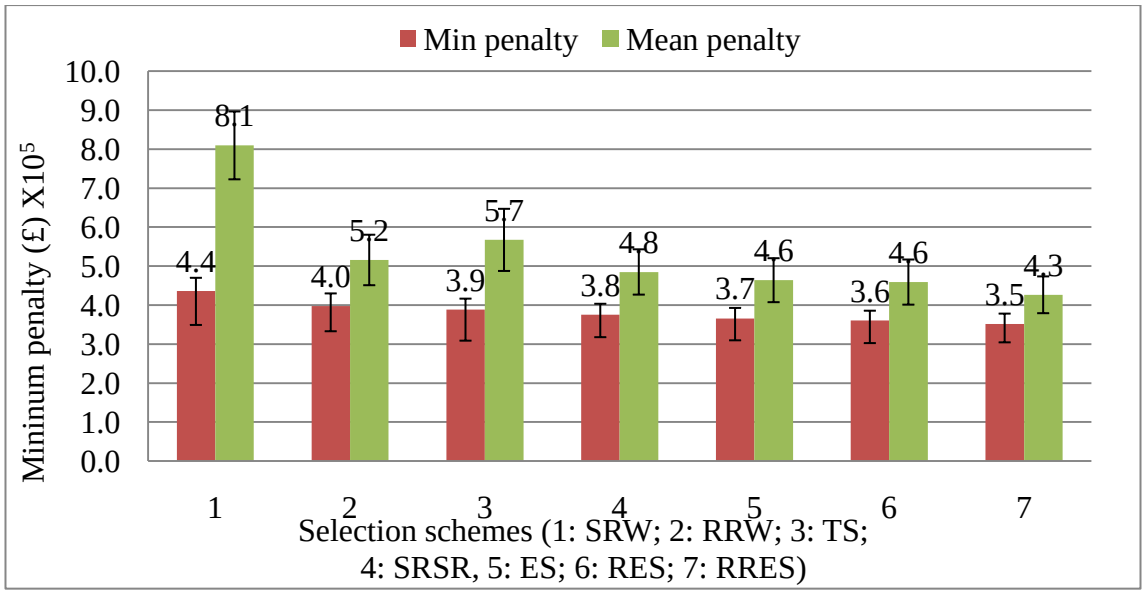

Figure 4-3. Performance comparison of 7 selection schemes

4.5 An Evaluation of Different Combination of Crossover and Mutation Operators

The next experiments evaluated the performance of different combinations of crossover and mutation operators in order to determine the optimal configurations to solve the full size problem. The extra large problem was considered as it had a large search space. The probability of crossover chosen was 0.8 and the probability of mutation chosen was 0.08 which were values used by Pongcharoen (2001). Each run with a specified configuration was replicated 50 times. The results obtained by the various combinations of crossover and mutation operators are shown in Table 4-5.

The best and worst combinations are shown in bold with the best with an underline. The best combination is PBX*2OAS and the worst combination was 1PX*CIM. There were significant differences between the best and worst results. The best combination achieved cost reduction of 28.2% compared to the worst. This indicates that the importance of choosing an appropriate combination of genetic operators. For each crossover operator the result from the best combination is shown underlined, whilst for mutation operator the result is shown stroked through. A mean result is also provided for each operator. The operator with the best mean is shaded. The crossover operators that had the best mean performance were OX. The mutation operators that had the best mean performance were SM. Results suggested that operators that exchange random or small length of strings such as 2OAS and OX performed better than those that exchange fix length of strings such as CIM. The results also suggested that operators such as OX, MPX that transfer ordering information from parents to children, perform better than operators that are ineffective as transferring ordering information such as AEX.

	CIM	2OAS	3OAS	2ORS	3ORS	IM	DM	SM	mean
1PX	***379413***	339931	330642	323558	317282	314239	312391	303326	327598
AEX	302842	302414	302240	301551	301551	301397	300746	300746	301686
CYX	299294	290469	289286	286953	285672	284370	282065	280239	287293
ERX	280239	280239	280239	280239	280239	280239	280239	280239	280239
MPX	278594	277867	277851	277602	275970	275961	275884	275301	276879
OX	275301	275301	~~275301~~	~~275301~~	~~274904~~	~~274904~~	~~274904~~	~~274904~~	275102
PBX	~~274720~~	***272591***	375946	338025	329662	322578	316301	313258	317885
PMX	311411	302166	301682	301254	301080	300392	300392	300237	302327
2PCX	299587	299587	298134	290272	289089	286756	285475	284173	291634
mean	300156	293396	303480	297195	295050	293426	292044	290269	

Table 4-5 Performance of combinations of crossover and mutation operators

4.6 Solving a Full size Industrial Problem with ESOGAST

An 18 months production schedule was obtained from the Heavy Machine Shop at a collaborating capital goods company that manufactured large steam turbine generators. The facility had 52 machine tools (although only 36 were used during the period) and manufactured the highest cost items within the product (e.g. turbine rotors and stator casings). The product structure information was also obtained, which provides information on assembly relationships. Compared to other industrial problems, it can be seen in Table 4-1 that the full company schedule is a significantly larger problem than the other production scheduling problems previously considered by Pongcharoen (2001). The company did not estimate the duration of assembly processes. Information on the

capacity of assembly areas was also unavailable. A conservative assumption was that each assembly took one month to complete under infinite capacity conditions. These values were chosen on the basis of conversations with practitioners conducted by Hicks (1998).

The ESOGAST was used to find the optimum schedule for the full schedule problem. Two cases were considered with best and worst configuration for the ESOGAST base on previous experiment. The configurations were shown in Table 4-6. It can be easily seen in Figure 4-4 that the performance of case 2 is significant better than that of case 1. The trend of the value of case 2 is preferable as it converges steadily and quickly whereas in case 1, the value fluctuates and does not converge. This emphasises that the configuration has large impact on the performance of Genetic Algorithms.

Configuration	Case 1 (worst)	Case 2 (best)
Population size	200	200
Generation number	100	100
Crossover operator	1PX	PBX
Crossover probability	0.8	0.8
Mutation operator	CIM	2OAS
Mutation probability	0.08	0.08
Selection Scheme	SRW	RRES

Table 4-6. Configuration of the ESOGAST for full schedule problem

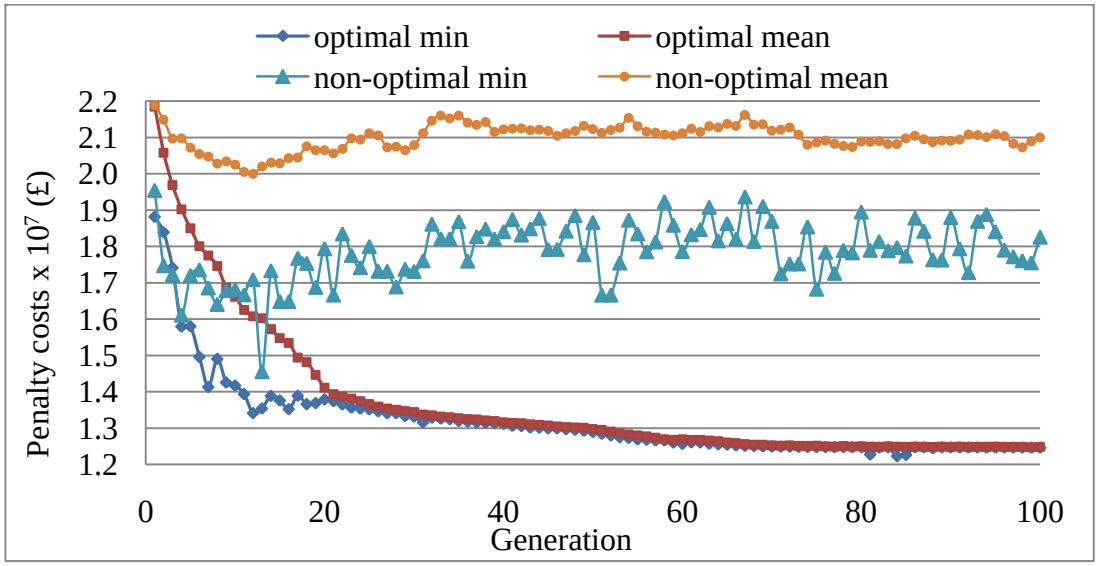

Figure 4-4 Minimum penalty for the full size schedule

In this problem, the search space is extremely large. Increasing the population size increases the amount of search within each generation. The computational time required to solve the full size problem was about 1.5 and 5 hours with optimal and non-optimal

configurations respectively. It is estimated that GAST would take approximately 3 to 9 months for a single run based upon previous experiment. The CPU time of the GAST for the full size problem hence is considered unrealistic and the GAST would not be a practical tool for the company. However, the CPU time of ESOGAST demonstrates that it provides a practical tool for solving very large industrial scheduling problems which are commonly encountered in capital goods companies.

4.7 Summary

The Enhanced Single-objective Genetic Algorithm Scheduling Tool (ESOGAST) has been developed to tackle extremely complex production scheduling problem in capital goods industry. This ESOGAST consists of the following elements: input/output, graphical user interface, genetic representation, population initialisation, genetic operations, repair processes, fitness assignment and measurement, genetic selection scheme, parameter settings and stopping criteria. A new, more efficient repair processes was developed to rectify infeasible schedules encountered when optimising very large problems. This made it possible to solve a full size problem that used an 18 month schedule from a participating company with 52 machines.

The ESOGAST was compiled in C, whereas GAST used the interpreted language Tk-Tcl. Tests found that new tool was more than 1,300 times faster. This made it possible for ESOGAST to use larger populations with more generations, which considerably increased the amount of search. It also benefitted from the development of better selection schemes that were developed in this research. The quality of solutions produced by ESOGAST was compared with GAST. It produced schedules that reduced the minimum penalty cost by 5.3%, 31%, and 46% for the small, medium and large problems respectively.

The Selection Scheme had a large impact on the performance of Genetic Algorithm. It was found that biased mechanisms such as Elitist Strategy performed much better than unbiased mechanisms that used random selection (e.g. the Roulette Wheel). Combining selection schemes proved to be an even more effective strategy; the Elitist Strategy with Rank-based Roulette Wheel performed best.

The combination of crossover and mutation operators used had a large impact on performance. Operators that maintained the ordering information from parents

performed better than those that did not. Operators that performed best exchanges string of random length, whereas operators that exchanged fixed length strings produced relatively poor results. The results were particularly poor if the substring transferred half the length of the chromosomes or more.

The ESOGAST was successfully applied using optimal configurations based on previous experiments to solve full schedule problem using an 18 month schedule from a facility with 52 machines within reasonable time (1.5 hours). Results also showed that ESOGAST with optimal configurations is much more superior to that with non-optimal configurations.

Chapter 5. Development of a Multi-Objective Genetic Algorithm Scheduling Tool

This chapter describes the development of a Multi-Objective Genetic Algorithm Scheduling Tool (MOGAST) for scheduling complex products in the capital goods industry with multiple resource constraints and deep product structure. The MOGAST consists of the following elements: input/output, graphical user interface, genetic representation, population initialization, genetic operations, repair processes, fitness assignment and measurement, genetic selection scheme, parameter settings and stopping criteria. These elements are described individually. A detailed specification of the tool is provided.

5.1. Overview

In this chapter, a Multi-Objective Genetic Algorithm Scheduling Tool (MOGAST) was developed to tackle multiple objectives production scheduling problems. The MOGAST maintains most of the elements of ESOGAST (e.g. the GA elements and the repair process), but has a different fitness measurement and assignment scheme in order to find Pareto optimal solutions (Deb, 2004). The MOGAST program was also written in C (Kernighan and Ritchie, 1988).

The MOGAST was designed to maximise delivery performance and to reduce inventory whilst simultaneously maximising resource utilisation. These are conflicting objectives in scheduling problems. The delivery performance is represented by penalties which are caused by early or late delivery of final products to customers. The inventory arises from raw materials, work-in-progress and finished goods stocks. The early completion of components and assembly parts gives rise to additional work in progress. The machine utilisation is the percentage of time that the machines are used.

5.2. Elements Development of a Multi-Objective Genetic Algorithm Scheduling Tool

The main distinction between GAs and multiple objectives GAs is the way fitness is measured and assigned to individuals. The MOGAST algorithm developed in this work was based on that of MOGA develop by Fonseca and Fleming (1993) which was described in section 2.5.4 of Chapter 2. The MOGAST program was developed based on the Enhanced Single-Objective Genetic Algorithm Scheduling Tool (ESOGAST)

proposed in Chapter 3 by the author. This MOGAST consists of GA elements of gene encoding, genetic representation, population initialization, genetic operations, repair processes, fitness assignment and measurement, selection scheme and stopping criteria. The algorithm is illustrated in Figure 5-1. The only difference between ESOGAST and MOGAST is fitness measurement and assignment which is circled in grey in the figure.

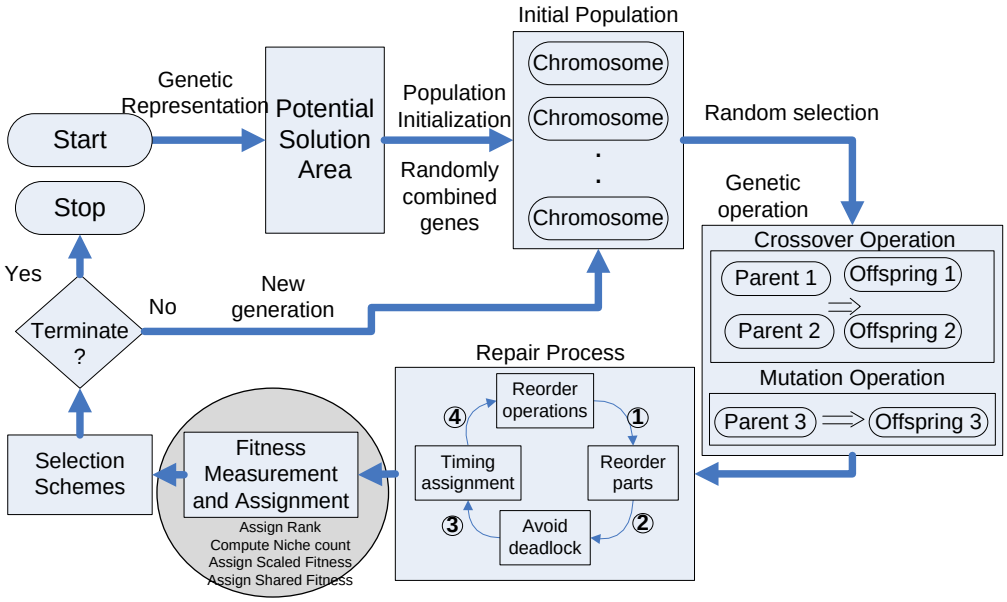

Figure 5-1 Structure of MOGAST for production scheduling

The coding schemes for product structure, genetic representation and population initialisation are same to those of ESOGAST described in section 3.2 of Chapter 3. The main difference between ESOGAST and MOGAST is process of fitness measurement and assignment and it is described in the following section.

5.2.1. Fitness assignment and measurement

After population initialisation, genetic operation and repair process, the next step is to perform the fitness measurement and assignment procedure. Steps are detailed as following (Deb, 2004):

Step 1.Set counter $i = 1$. Choose a sharing parameter σ_{share}. Initialise $\mu(j) = 0$ ($j = 1,\ldots, N$) for all ranks (here parameter σ_{share} is known as niche size, μ is the number of solutions within a same rank. Both are described in Chapter 2);

Step 2.Compute the number of solutions (n_i) that dominate solution i. Compute and assign the rank of solution i as $r_i = 1 + n_i$. Increase the number of solutions in rank r_i by one, so that $\mu(r_i) = \mu(r_i) + 1$;

Step 3. Repeat step 1 and step 2 until $i < N$ (total number of solutions). Otherwise, go to step 4;

Step 4. Set a rank counter r = 1. Identify the maximum rank r^* by checking the largest r_i which has $\mu(r_i) > 0$. Assign average fitness to any solution $i = 1,...,N$ from the best rank 1 to worst rank $n \leq N$ using a linear function (Deb, 2004):

$$F_{i(avg)} = N - \sum_{K=1}^{r_i-1} \mu(k) - 0.5(\mu(r_i) - 1) \quad \text{Equation 5-1)}$$

Step 5. Calculate the niche count for each solution i in rank r_i with other solutions of the same rank by using Equation 5-2.

$$nc_i = \sum_{j=1}^{\mu(r_i)} Sh(d_{ij}) \quad \text{Equation 5-2)}$$

In Equation 5-2, d_{ij} is the normalised distance between any two solutions i and j in a rank and is calculated as follows:

$$d_{ij} = \sqrt{\sum_{k=1}^{M} \left(\frac{f_k^{(i)} - f_k^{(j)}}{f_k^{max} - f_k^{min}} \right)^2} \quad \text{Equation 5-3)}$$

Where f_k^{max} and f_k^{min} are the maximum and minimum objective function value of the k-th objective. Also in Equation 5-2, Equation 5-4 is used with α=1 to compute the sharing function value.

$$Sh(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{share}}\right)^{\alpha}, & \text{if } d \leq \sigma_{share}; \\ 0, & \text{otherwise;} \end{cases} \quad \text{Equation 5-4)}$$

Step 6. After calculating the niche count, the next is to calculate the shared fitness using $F_{i(share)} = F_{i(avg)} / nc_i$. To preserve the same average fitness, multiply $F_{i(share)}$ with scaling factor S (shown in Equation 5-5) so that the scale fitness value is the same as the original average fitness value.

$$S = F_{i(avg)} . \mu(r_i) / \sum_{k=1}^{\mu(ri)} F'_{k(share),} \quad \text{Equation 5-5)}$$

Now the scale fitness is shown as follows:

$$F_{scale} = F_{i(share)} . S, \quad \text{Equation 5-6)}$$

Step 7. If r < r*, increment r by one and go to step 5. Otherwise, terminate the process.

In step 2, in order to assign the rank, a dominance calculation is needed. Hence solutions are evaluated with the fitness functions. In this research, the aim was to minimise delivery penalty, inventory costs and simultaneously maximise resource utilisation by minimising the total idle time of machines. The three fitness functions were shown in Equation 5-7, 5-8 and 5-9. Note that the inventory costs of work-in-

progress before the final operation of each component are not considered. However, this is measured indirectly thorough *F3* to minimise the waiting times between operations. The larger the idle time of machines is the longer of waiting times between operations is.

$$F1 = delivery\ performance\ = P_t * \sum_{i=1}^{n} E_{pi} + P_e * \sum_{i=1}^{n} T_{pi} \quad \text{Equation 5-7)}$$

$$F2 = inventory\ costs = P_e * \sum_{i=1}^{n} (E_{ci} + E_{ai}) \quad \text{Equation 5-8)}$$

$$F3 = Total\ idle\ time\ of\ machines = \sum_{k=1}^{M} I_k \quad \text{Equation 5-9)}$$

Where

$T_{Pi} = \max(0, F_{pi} - D_{pi})$ $E_{pi} = \max(0, D_{pi} - F_{pi})$ $E_{ci} = \max(0, S_{pai} - F_{cci})$

$E_{ai} = \max(0, S_{pai} - F_{cai})$ $I_k = \sum_{i=1}^{N} IO_i$ $IO_i = SO_{i+1} - FO_i$

Notation: subscript *a, c, p, i* (assembly and sub-assembly, component, product, item number), *n* is the number of the according items (either assembly, or component, or product) and N is the number of operations. It is vary and depends on each machine.

$P_t = Penalty\ rate\ of\ tardiness\ (100\ pounds\ per\ day)$

$P_e = Penalty\ rate\ of\ earliness\ (50\ pounds\ per\ day)$

$T_{pi} = Tardiness\ of\ product\ (days)$ $E_{pi} = Earliness\ of\ product\ (days)$

$E_{ci} = Earliness\ of\ component\ (days)$ $E_{ai} = Earliness\ of\ assembly\ (days)$

$F_{pi} = Finish\ time\ of\ product\ (date)$ $D_{pi} = Due\ date\ of\ product\ (date)$

$S_{pai} = Start\ time\ of\ parent\ assembly\ of\ component\ i\ (date)$

$F_{cci} = Finish\ time\ of\ child\ component\ i\ (date)$

$F_{cai} = Finish\ time\ of\ child\ assembly\ i\ (date)$

$I_k = Idle\ time\ of\ machine\ k\ (hour)$ $IO_i = Idle\ time\ of\ i^{th}\ operation\ (hour)$

$FO_i = Finish\ time\ of\ i^{th}\ operation\ (date)$

$SO_{i+1} = Start\ time\ of\ (i+1)^{th}\ operation\ (date)$

5.3. Specification of the MOGAST Model

This section provides a functional specification of MOGAST and summarises its construction and technical features. A flowchart of MOGAST is also provided.

5.3.1. General principles and data structure

The same general principles and data structure are applied as described in section 3.3.1 and 3.3.2 of Chapter 3 for ESOGAST.

5.3.2. Input/output files

This MOGAST used the same input/output data files for schedules produced as described in section 3.3.3 of Chapter 3 for ESOGAST. However there was a small difference of the summary file in which information is shown as listed by the following:

1. The file uploaded in this run;
2. The configuration of the MOGAST applied in this run;
3. The inventory cost, delivery performance, idle time of machines and total costs of Pareto optimal schedules in each generation in a particular configuration;
4. The minimum, mean and maximum penalty and standard deviation of each population;
5. The inventory cost, delivery performance and idle time of machines of Pareto optimal schedules of different replications in a particular configuration;
6. The inventory cost, delivery performance and idle time of machines of Pareto optimal schedules of all replications in all configurations;
7. The total execution time of each run.

5.3.3. Functional specification of the MOGAST program

The sequence of the MOGAST procedure is as follows:

1) Parameter setting: the program enables the executor to input the parameters as wished. These parameters are: problem size, sigma value (σ), alpha value (α), population size, chromosome length, number of generation run, genetic selection scheme, crossover operator, crossover probability, mutation operator, mutation probability; (go to 2);
2) Population initialisation: there are two steps to initialise population.
    a) A string of genes (chromosome) is generated and stored, which read and contains the data information from the input file. This single chromosome is the initial chromosome to create a population and it uses double linked list structure. (go to 2b);
    b) A gene is randomly selected from the sub-chromosome within the initial chromosome. Genes are then copied to generate a list of genes to produce sub-chromosome and chromosome. (go to 2c);

c) Step 2b is repeated until the number of genes and number of chromosomes generated equal to the specified population size. Each gene is double linked with each other and each chromosome is also double linked with each other. The initial population is then produced. (go to 3);

3) Genetic operations: The genetic operations procedures that used for ESOGAST and described in section 3.3.5 are applied here. (go to 4);
4) Repair process: The repair process that used for ESOGAST and described in section 3.3.5 is applied here. (go to 5);
5) Fitness evaluation: three fitness functions are used to evaluate the performance of each chromosome respectively in order to calculate the dominance relationships among chromosomes and then assign the rank base on the fitness value regarding to each fitness function. It has the following sequences:
   a) Regarding to the delivery performance, the specified due date for each final product is compared with its completion time. If final products are finished late, lateness time for all products is summed. If final products are finished early, the earliness time for all products is summed. The penalty cost associated with the schedule can be determined from the lateness or earliness times as described on Equation 5-7. Earlier completion time for each component, assembly and subassembly is compared with its due date and is accumulated. The total inventory cost associated with the schedule can be determined by Equation 5-8. The idle time between operations of the same machine are accumulated which is the idle time of this machine during actual production scheduling process. Then the idle times of all machines are also accumulated and can be determined by Equation 5-9. (go to 5b);
   b) Step 5a is repeated for all chromosomes. (go to 5c);
   c) Compare the fitness among chromosomes regarding to three objective functions respectively, calculate the number of chromosomes dominating each chromosome and assign the rank to each chromosome. (go to 5d);
   d) Assign the raw average fitness to each chromosome from rank 1 to rank N. (go to 5e);
   e) Calculate niche count for chromosomes from rank 1 to rank N. (go to 5f);
   f) Calculate shared fitness for chromosomes from rank 1 to rank N. (go to 5g);
   g) Calculate scaled fitness for chromosomes from rank 1 to rank N. (go to 5h);
   h) The best chromosomes with rank 1 in this generation are updated and stored. (go to 6);

6) Chromosome selection: based on the selection scheme chosen in step 1), the selection schemes are applied accordingly. The successive chromosome is selected and placed in the initial population. This procedure repeats until the desired population size is met and the previous initial population is replaced. (go to 7);
7) Result: the chromosome with minimum penalty is stored. The minimum, mean and maximum penalties of a population are also stored. (go to 8);
8) Program termination: termination condition is checked. If the condition is satisfied, go to 8. Otherwise, next generation is required. (go to 3);
9) Displaying the result: two output file are produced. (go to 10);
10) The MOGAST is terminated.

Figure 5-2 demonstrates the flow chart diagram of the MOGAST programme.

5.3.4. Assumptions within the MOGAST model

This MOGAST will use the same assumptions described in section 3.3.6 of Chapter 3 for ESOGAST.

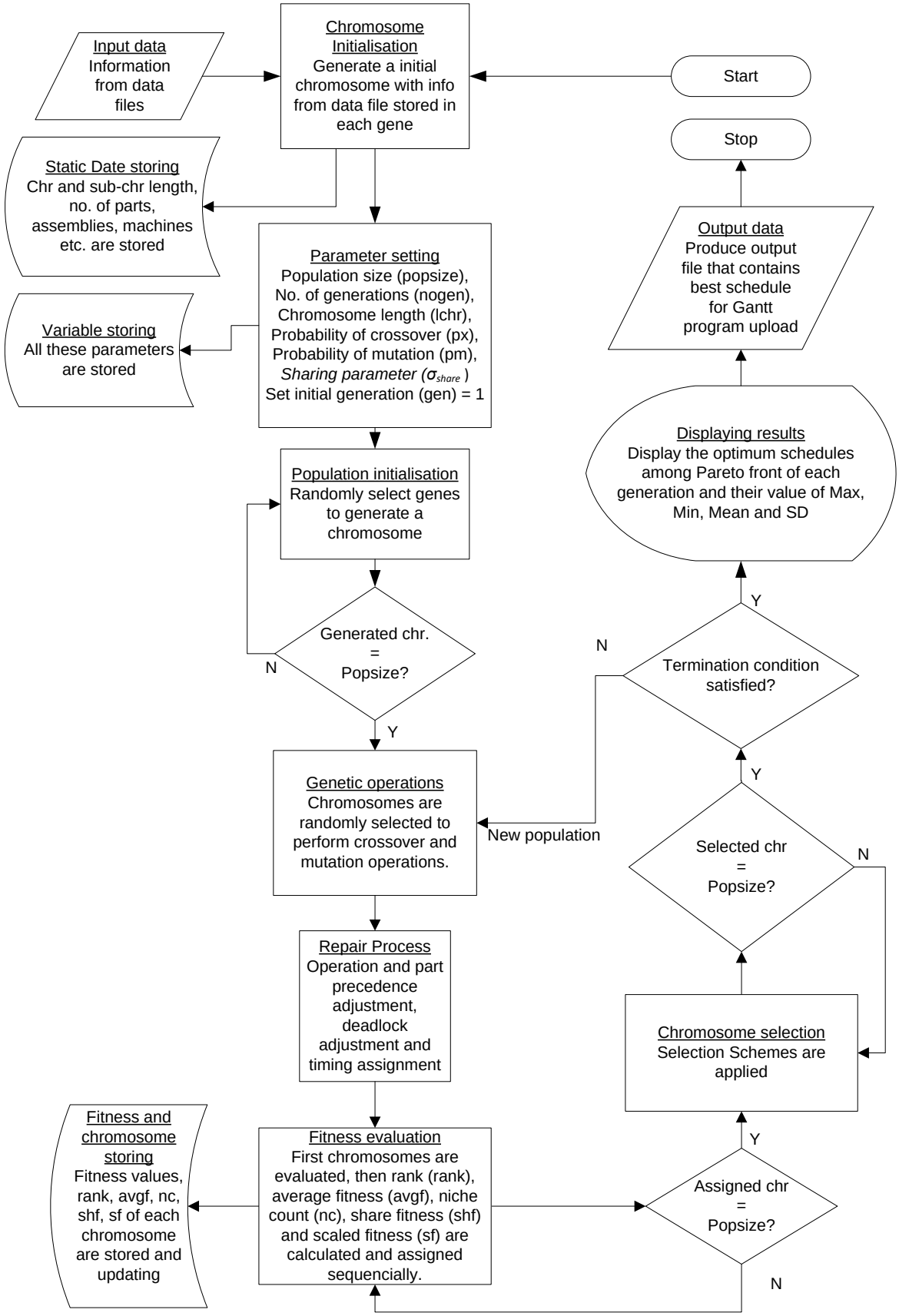

Figure 5-2 Flow chart of the MOGAST program

Chapter 6. Experimental Programme II

This chapter describes the experimental programme that utilised the MOGAST. There are seven experiments. The first five experiments focused on problems with two objectives which were to maximise delivery performance and minimise inventory cost. The objectives of these experiments were to: 1) determine the best parameter settings of crossover probability and mutation probability; 2) identify the best combination of genetic operators for the MOGAST and compared to previous work; 3) identify the best selection schemes for the MOGAST and compare to previous work; 4) evaluate the impact of niche size (σ) on the performance of the MOGAST; 5) to apply the MOGAST to solve a full size industrial scheduling problems using the best configuration of the MOGAST based on the previous four experiments. The last two experiments focused on problems with three objectives and the objectives of these experiments were to: 1) evaluate the trade-offs amongst delivery performance, inventory cost and resource utilisation; 2) apply the MOGAST to solve a full size industrial schedule with objectives that maximise delivery performance, minimise inventory costs and maximise resource utilisation.

6.1 Experimental Design

The medium, large, extra large and full size schedule shown in Table 4-1 were considered in the following experiments. The information relating to the four industrial problems were summarised in Table 4-1 in Chapter 4. The GA population sizes chosen were 40, 60, and 80 for medium, large, extra large respectively with 20 generations (as used by Pongcharoen (2001)). A population size of 200 with 100 generations was chosen for the full size schedule based on previous experiment in Chapter 4. The value $\sigma_s = 0.5$ was used for the experiments in section 6.2, 6.3, and 6.4 (as recommended by Deb (2004)). For medium, large and extra large problem that studied in section 6.2 to section 6.5, each run of a specified configuration was replicated fifty times to facilitate a comprehensive statistical study with consideration of a realistic computation time.

6.2 Determine the Best Crossover and Mutation Probability

This experiment aimed to identify the best crossover and mutation probabilities for the MOGAST. Crossover probabilities (CP) in the range of 0.6~0.9 and mutation probabilities (MP) in the range of 0.01~0.1 were evaluated (as recommended by Todd (1997)). The position based crossover (PBX) crossover, two operations adjacent swap

(2OAS) mutation and rank-based roulette-elitist strategy (RRES) were found to perform best for the ESOGAST. Hence in this experiment, these genetic operators were used. The results are shown in Table 6-1. There are sixteen combinations of different crossover and mutation probabilities. The numbers in the table indicate the number of Pareto-optimal solutions for each of the sixteen combinations along the Pareto front. It can be seen that the combination of 0.8*0.01, 0.8*0.01 and 0.9*0.01 had the largest number of Pareto solutions (199, 44 and 19 which are shown in italic bold) and hence these were the best combinations for medium, large and extra large problems respectively.

Medium problem		Crossover probability 0.6	0.7	0.8	0.9
Mutation Probability	0.01	28	182	***199***	100
	0.02	68	19	182	14
	0.05	124	150	47	9
	0.1	104	67	124	83
Large problem		Crossover probability 0.6	0.7	0.8	0.9
Mutation Probability	0.01	0	0	***44***	19
	0.02	2	10	0	13
	0.05	0	14	4	0
	0.1	0	34	3	0
Extra large problem		Crossover probability 0.6	0.7	0.8	0.9
Mutation Probability	0.01	0	4	1	***19***
	0.02	3	1	0	8
	0.05	1	0	10	0
	0.1	0	0	1	2

Table 6-1. Number of Pareto solutions with different crossover and mutation probabilities

6.3 An Evaluation on Performance of the Different Combination of Genetic Operators

In this experiment, the performance of different combinations of crossover and mutation operators was compared. Based on the previous experiment, the best combinations of crossover and mutation probabilities of 0.8*0.01, 0.8*0.01 and 0.9*0.01 were chosen for the medium, large and extra large problems respectively. The rank-based roulette-elitist (RRES) was superior to other strategies in the previous experiment for ESOGAST, hence was used in this experiment.

In this experiment there were 72 results for each of the combinations of crossover and mutation operators for each problem (9 crossover operators X 8 mutation operators as list in Table 3-1 and Table 3-2). The numbers in the table indicate the number of Pareto-optimal solutions for each combination along the Pareto front. It can be seen from Table 6-2 that for the medium problem, PBX and IM was the best combination, for the large problem, CYX and IM was the best combination and for the extra large problem, the best combination was CYX and 3ORS. It can be also observed based on the mean values that those crossover operators that performed best were PBX, PMX and CYX. The mutation operators that perform best were 2OAS, 3OAS, SM, 3ORS and IM.

Medium problem		Crossover operators									
		1PX	AEX	CYX	ERX	MPX	OX	PBX	PMX	2PCS	Mean
Mutation operators	CIM	125	2	0	0	17	41	241	106	0	59.1
	2OAS	38	1	6	10	25	12	162	119	0	41.4
	3OAS	138	6	101	0	25	25	222	182	5	78.2
	2ORS	40	1	69	0	31	62	119	75	0	44.1
	3ORS	136	1	40	0	35	8	210	126	2	62
	IM	7	2	0	8	13	0	***265***	82	0	41.9
	DM	161	5	70	0	24	8	151	140	0	62.1
	SM	81	5	89	0	26	45	201	117	22	65.1
	Mean	90.8	2.9	46.9	2.3	24.5	25.1	196.4	118.4	3.6	
Large problem		Crossover operators									
		1PX	AEX	CYX	ERX	MPX	OX	PBX	PMX	2PCS	Mean
Mutation operators	CIM	0	1	2	0	0	0	3	27	0	3.7
	2OAS	0	0	0	0	0	4	0	6	0	1.1
	3OAS	29	0	0	0	0	0	0	3	0	3.6
	2ORS	0	0	60	0	0	0	0	12	0	8
	3ORS	0	0	0	0	0	0	50	37	0	9.7
	IM	0	0	***120***	0	0	0	0	0	0	13.3
	DM	0	0	0	0	0	0	0	0	0	0
	SM	0	0	0	0	0	0	0	0	0	0
	Mean	3.6	0.1	22.8	0	0	0.5	6.6	10.6	0	
Extra large problem		Crossover operators									
		1PX	AEX	CYX	ERX	MPX	OX	PBX	PMX	2PCS	Mean
Mutation operators	CIM	0	0	0	0	0	0	0	2	0	0.22
	2OAS	0	0	0	0	0	0	4	0	0	0.44
	3OAS	0	0	0	0	0	0	4	0	0	0.44
	2ORS	0	0	0	0	0	0	0	0	0	0
	3ORS	0	0	***12***	0	0	0	0	0	0	1.33
	IM	0	0	0	0	0	0	2	0	0	0.22
	DM	0	0	0	0	0	0	0	0	0	0
	SM	0	0	0	0	0	0	1	0	0	0.11
	Mean	0	0	1.5	0	0	0	1.38	0.25	0	

Table 6-2. Number of Pareto solutions with different combination of crossover and mutation operators

6.4 An Evaluation on Performance of Different Selection Schemes

In this experiment, the performance of seven different selection schemes was evaluated. The crossover and mutation probabilities adopted were again based on the experiment described in section 6.2. The combinations of crossover and mutation operators for the medium, large and extra large problem were based on the experiment outlined in section 6.3. The results are shown in Table 6-3. It shows that SRSR, RES and RRES were the best selection schemes for the medium, large and extra large problems respectively. Results also suggested that the Roulette Wheel and Elitist combined strategy generally performed best. In Figure 6-1, dominated and non-dominated solutions are shown, the Pareto front are along the left hand side of the cluster of points. The points to the right of the Pareto front are dominated solutions.

	Selection Schemes						
	SRW	RRW	TS	SRSR	ES	RES	RRES
Medium problem	210	86	9	***322***	58	311	203
Large problem	73	0	0	0	5	***98***	0
Extra large problem	1	0	0	4	0	30	***72***

Table 6-3. Number of Pareto solutions with different Selection Schemes

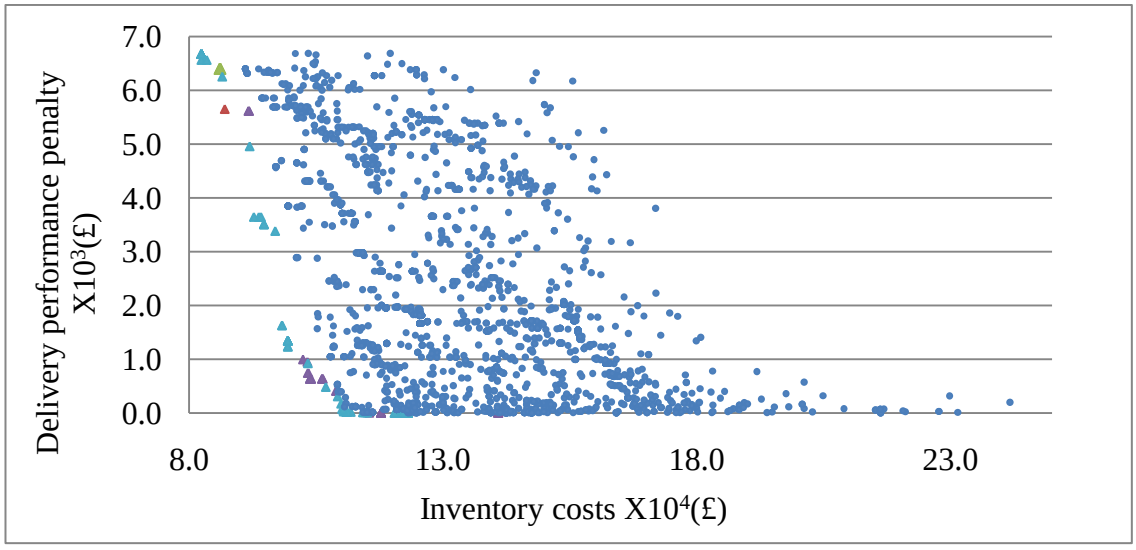

Figure 6-1. Dominated (triangle dots) and non-dominated (round dots) solutions obtained with different Selection Schemes (extra large problem)

6.5 An Evaluation on Performance of Various Sigma (niche size) Settings

In section 2.5.2 of chapter 2, the importance of the niche size was explained. The literature has generally overlooked the impact of σ_s on the performance of MOGAs. In this experiment, an evaluation was carried out to determine the best value of the niche size (σ_s). Deb (2004) recommended niche sizes in the range 0 to 1. Hence, the values of

niche size evaluated in this experiment were 0.01, 0.05, 0.1, 0.3, 0.5, 0.7, 0.9 and 1.0. Although it is not realistic to investigate all the values within the range, these chosen values are generally spread within the range.

The results are shown in Table 6-4. It can be seen that the best niche size for medium, large and extra large problem were 1.0, 0.9 and 0.05. The results suggest that as the problem becomes bigger (and the search space gets larger), the MOGAST performs better with a smaller niche size. This may indicate that as the search space becomes larger, there are more potential solutions along the Pareto front and the distance among these solutions may be very close. Hence a small niche size may achieve better results. In Figure 6-2, the spread of non-dominated solutions is shown.

	Niche Size σ_s							
	0.01	0.05	0.1	0.3	0.5	0.7	0.9	1.0
Medium problem	83	200	224	314	276	374	373	***378***
Large problem	0	0	0	0	0	0	***113***	45
Extra large problem	27	***34***	25	0	13	12	0	0

Table 6-4. Number of Pareto solutions with different niche sizes

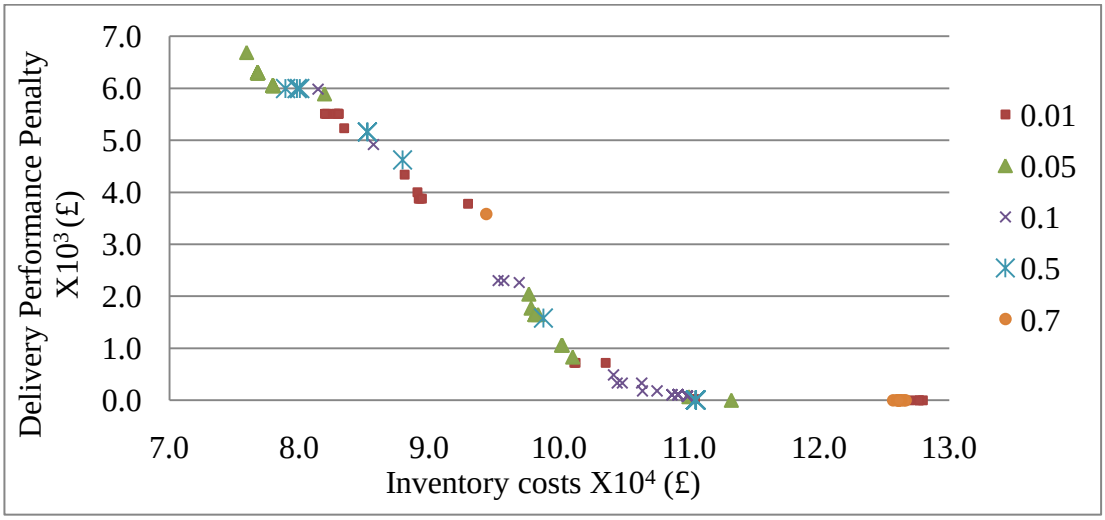

Figure 6-2. Pareto-optimal solutions obtained with different niche sizes (extra large problem)

6.6 Apply the MOGAST to Solve a Full Schedule Industrial Problem with Optimum Parameters based upon the Previous Experiments

The previous experiments helped identify the best GA configuration for each problem size. Based on these results, the MOGAST was applied to solve a very large real world problems that was based upon a full 18 month schedule of a product with 1070 different types of parts and 2720 components and assemblies processed on 36 machines (there

were also 16 unused machines). As the size of the extra large problem is closer to the size of the full schedule than other problems, the best and worst GA configurations for the extra large problem were used as the optimal and non-optimal configuration for the MOGAST to solve the full schedule problem. The configurations are shown in Table 6-5. The Pareto-optimal solutions obtained with optimal and non-optimal configurations were compared and shown in Figure 6-3 and Figure 6-4.

	Optimal configuration	Non-optimal configuration
Population size	200	200
Number of Generations	100	100
Crossover probability	0.9	0.9
Mutation probability	0.01	0.05
Crossover operator	CYX	1PX
Mutation operator	3ORS	IM
Selection schemes	RRES	SRSR
Niche size	0.05	0.1

Table 6-5. Configurations of the MOGAST for the full schedule problem

Figure 6-3 shows the Pareto-optimal solutions as the Pareto fronts in generation 1, 10, 30, 50, 70 and 100 respectively. As can be seen from this figure, the solutions in later generations were much better than those in previous generations. It also can be seen from the added polynomial trend lines that the Pareto-optimal solutions of each of the generations shown were approaching and converged to the Pareto front achieved by the last generation. This suggested that the GA configurations were set properly and the MOGAST achieved satisfactory results.

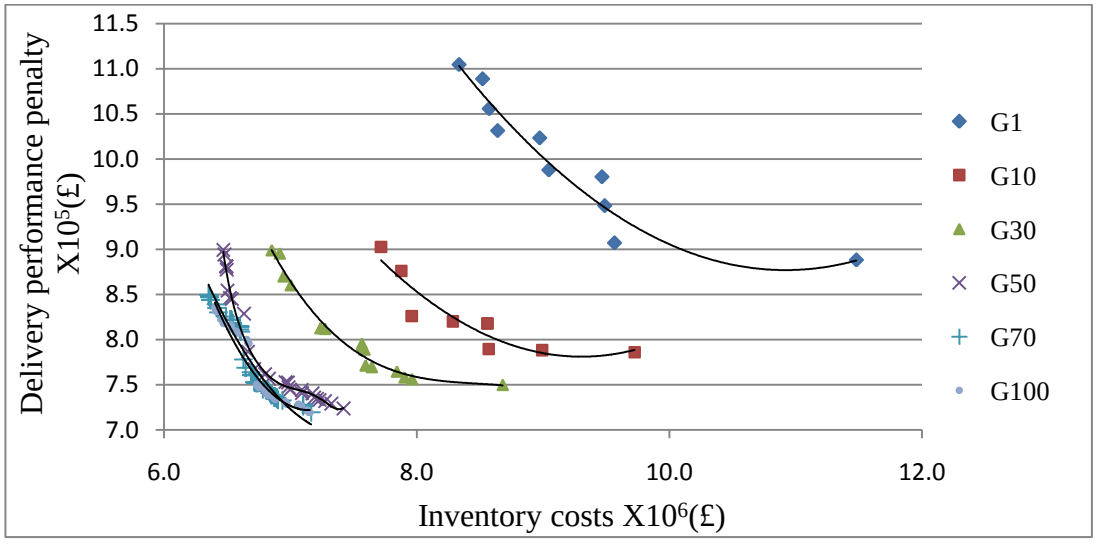

Figure 6-3 Pareto front results for full size problem with optimal configuration

The performance of the MOGAST with non-optimum configuration is shown in Figure 6-4. Although it also approached and converged to the Pareto front of last generation, it can be seen in Figure 6-5 that these Pareto-optimum solutions obtained by the last generation were inferior to those obtained with optimum configurations. This suggests that the results of the previous experiments were correct.

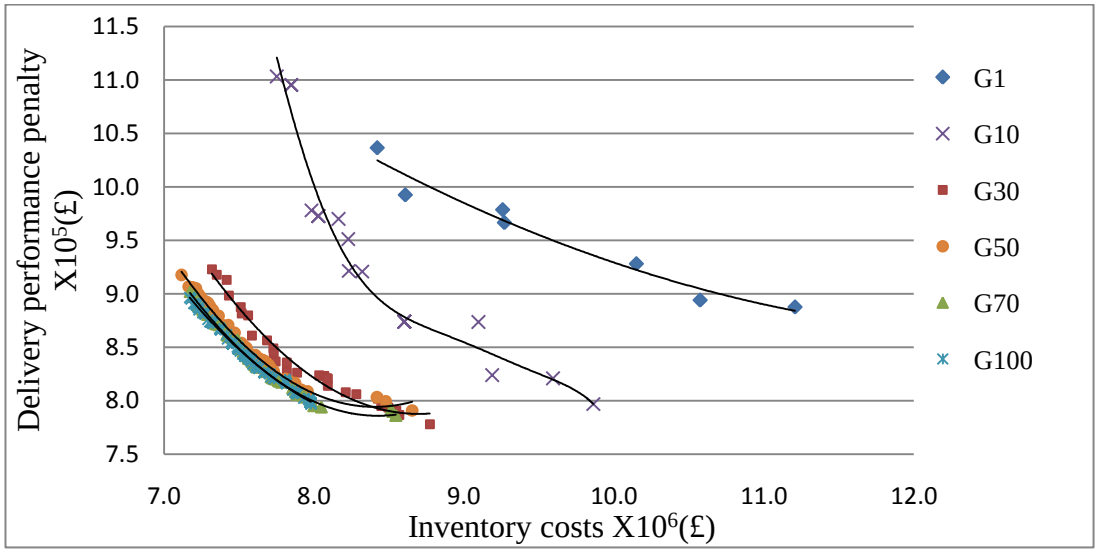

Figure 6-4. Pareto front results for full size problem with Non-optimal configuration

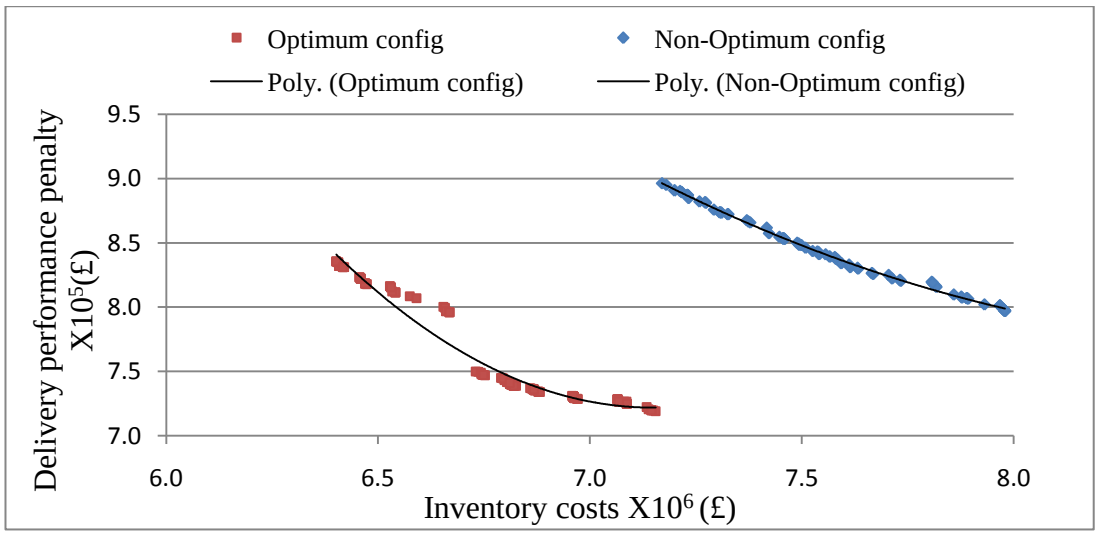

Figure 6-5. Performances with optimal and Non-optimal configuration

6.7 An Evaluation of Trade-off among Delivery Performance, Inventory Cost and Resource Utilisation

This experiment aimed to identify the existence of the trade-offs between delivery performance, inventory costs and machine utilisation. The extra large schedule was chosen because of its large search space. Optimal configurations were chosen for

MOGAST based on previous experiment and it is shown in Table 6-6. Figure 6-6 shows results of delivery performance against machine utilisation whilst Figure 6-7 shows the results of inventory cost against machine utilisation. In both figures, all the Pareto optimal solutions obtained of all fifty replicates are shown. It can be observed that both figures with emphasis by adding trendlines achieved trade-off. This suggests that the resource utilisation is trade-off with both the delivery performance and inventory cost. Similarly, trade-off between delivery performance and inventory cost was studied in previous experiments in this chapter. This experiment identified the conflicting nature of all three objectives of production scheduling in capital goods industrial and it also justified the necessity of providing such scheduling tool that is able to optimise all three objectives simultaneously for the decision makers.

Configurations	Value chosen
Niche size (σ)	0.05
Population size	80
Number of generations	20
Crossover probability	0.9
Mutation probability	0.01
Crossover operator	CYX
Mutation operator	3ORS
Selection scheme	RRES

Table 6-6. Configurations of MOGAST

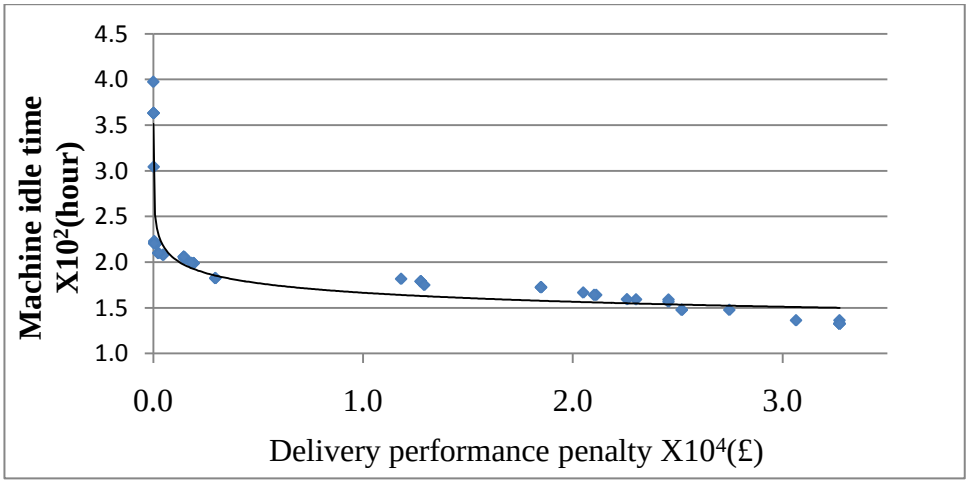

Figure 6-6. Trade-off between delivery penalty and machine utilisation

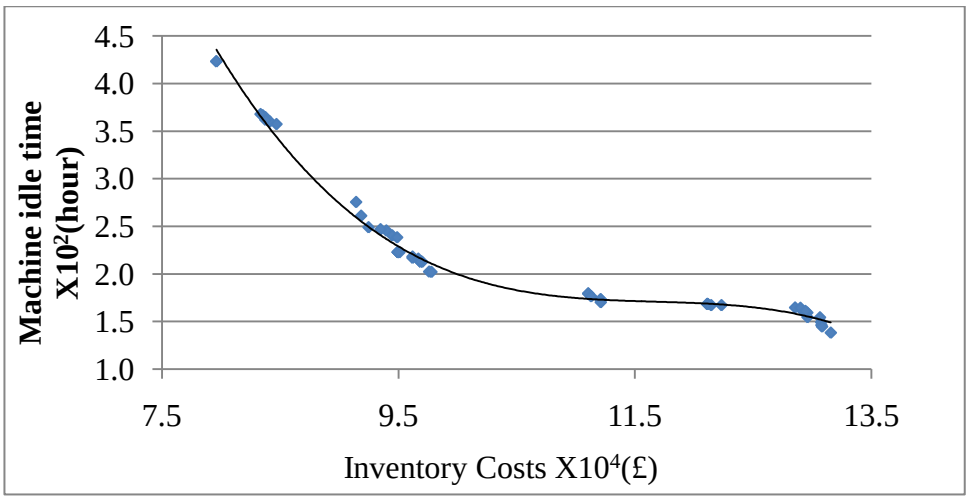

Figure 6-7. Trade-off between inventory cost and machine utilisation

6.8 Solve the Full Size Industrial Schedule with All Three Objectives Taken Into Consideration

This experiment was conducted to solve a full size scheduling problem from a capital goods company with all three objectives: maximising delivery performance, minimising inventory cost and maximising machine utilisation. The optimal configurations for extra large problem again were used in this experiment because of its large search space close to that of full schedule. The results are shown in Figure 6-8. The dots in this figure represent the Pareto optimal solutions with respect to all three objectives.

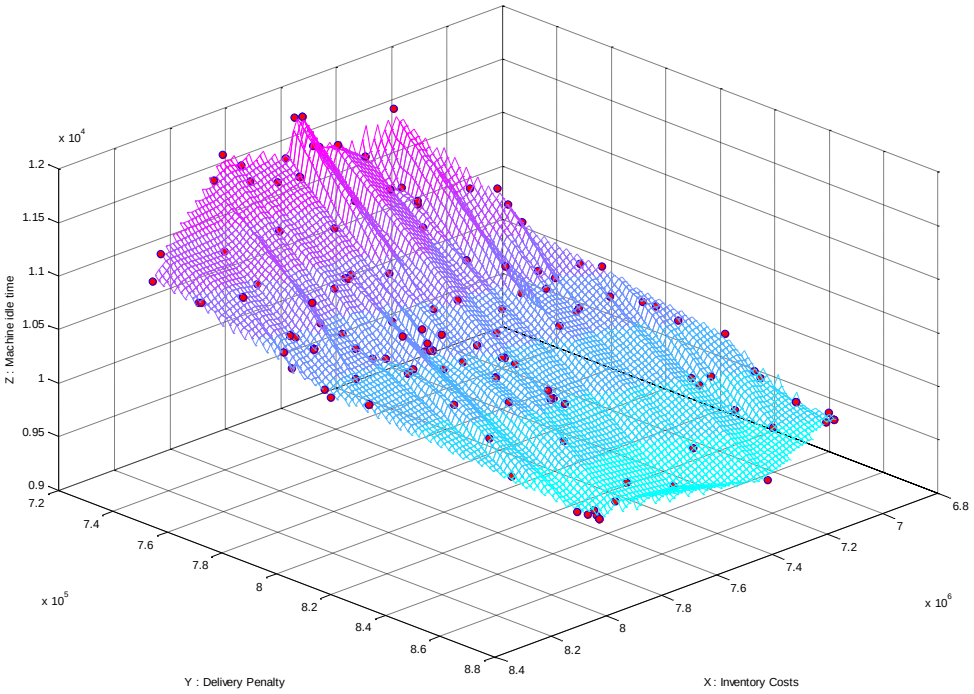

Figure 6-8. Pareto optimum solutions for full size schedule

The computation time of the run was 4979 seconds (83 minutes or 1.38 hours). The MOGAST was successfully applied to solve the full size scheduling problem from a capital goods company within a realistic time.

6.9 Summary

In this Chapter, the Multi-Objective Genetic Algorithm Scheduling Tool (MOGAST) was applied in order to optimise the delivery performance to meet customer requirement, while simultaneously reducing the inventory costs and maximise machine utilisation.

In section 6.2, 6.3, 6.4 and 6.5, a series of experiments was carried out to identify the best MOGA configuration, based upon a full factorial design. The experiment in section 6.2 aimed to identify the best crossover and mutation probability for the GA. It found out that the best combination of crossover and mutation probability for the medium, large and extra large problems are 0.8*0.01, 0.8*0.01 and 0.9*0.01. The experiment in section 6.3 identified that the crossover operators PBX, PMX, CYX were superior to AEX, ERX, MPX, and OX. The mutation operators 3ORS, 3OAS and IM were superior to other operators. However, the advantage of these mutation operators was not practically significant for all three problems studied. The experiment in section 6.4 identified that SRSR, RES and RRES were the best selection strategies for medium, large and extra large problem respectively. It also found out that the selection schemes that combined the Roulette Wheel and Elitist strategy performed better than schemes that did not. This result is similar to previous single criteria study in Chapter 4. The experiment in section 6.5 identified the impact on the performance of the MOGAST using various niche sizes. The best niche size for medium, large and extra large problems are 1.0, 0.9 and 0.05. The result suggests that as search space becomes bigger, a smaller value of niche size σ_s is preferable to produce the better results. The experiment in section 6.6 applied the MOGAST to solve a full size industrial scheduling problem with the optimal and non-optimal GA configuration based on the previous experiments. The MOGAST with both configurations obtained solutions approaching the Pareto front and the MOGAST performed better with the optimal configuration than that with the non-optimal configuration. The MOGAST achieved excellent results with the optimal configuration. The experiment in section 6.7 evaluated the existence of trade-off among delivery performance, inventory cost and resource utilisation. The extra large problem was considered. Results showed that all three objectives were trade-off with each other. The results justified the necessity of providing such a tool to optimise

all three conflicting objectives for production scheduling in capital goods industrials. The last experiment in section 6.8 applied the MOGAST to solve the full schedule problem with all three objectives taken into account. The experiment achieved satisfactory results and the computation time consumed was within a realistic time.

Chapter 7. Development of an Artificial Immune System Scheduling Tool (AISST)

This chapter describes the development of an Artificial Immune System Scheduling Tool (AISST) for scheduling complex products in capital goods companies. The AISST consists of the following elements: input/output, graphical user interface, parameter setting, antibody representation and initialisation, affinity measurement, clone strategy, mutation, antibody elimination, repair process and termination criteria. These elements are described individually. A detailed specification of the tool is provided.

7.1 Overview

In this work, an Artificial Immune System Scheduling Tool (AISST) was developed for solving production scheduling problems encountered in capital goods industry. The clonal selection theory (Burnet, 1959) was adopted for the development of this scheduling tool. This tool aims to produce optimal schedules by converting the planning information from a data file generated by the Manufacturing System Simulation Model developed by Hicks (1998). The AISST has the same objective of the ESOGAST, which was designed to minimise tardiness and earliness penalties caused by early or late arrival of components, assemblies and products.

7.2 Elements Development of an Artificial Immune System Scheduling Tool

Program of this tool is coded in C (Kernighan and Ritchie, 1988). This AISST also contains a repair process that was design to rectify illegal and deadlock schedules due to the product structure and finite resource capacity. Figure 7-1 shows the structure of the AISST, which includes parameter setting, antibody representation and initialisation, affinity measurement, clone strategy, mutation, antibody elimination, repair process and termination criteria. The proposed algorithm was developed base on the clonal selection and affinity maturation principles of the Artificial Immune System (AIS).

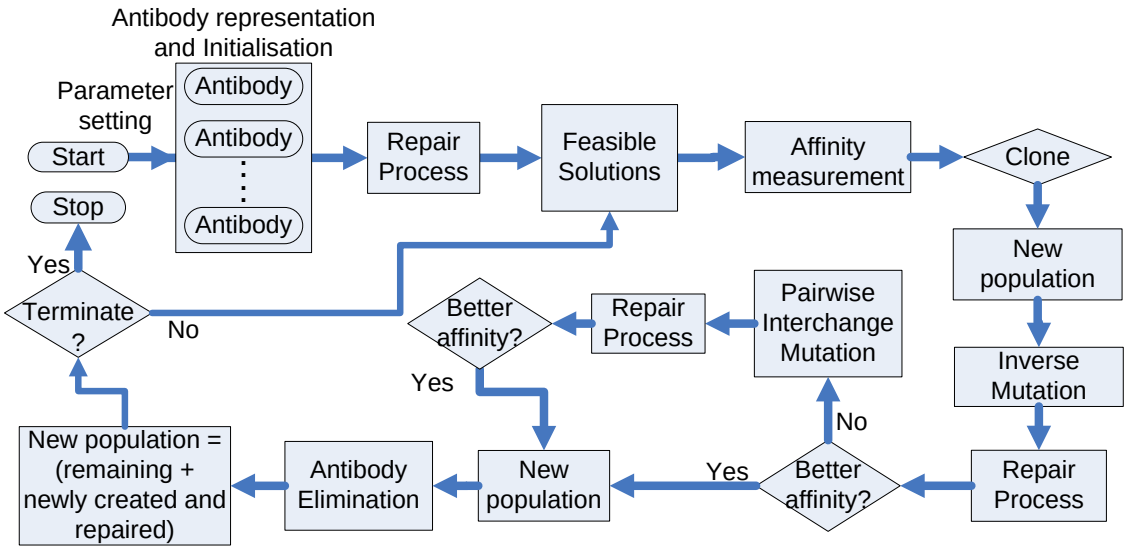

Figure 7-1. Structure of AISST for production scheduling

7.2.1 Product structure representation

The product structure representation of ESOGAST was used for the AISST.

7.2.2 Cell and antibody encoding

A common representation of the structure encoding for production scheduling problem is sequence of the jobs. Based on this fact, the sequence of operations within each production is encoded as a string. A whole antibody represents the complete sequences of all operations within a schedule. Because operations are operated on different non-identical machines, a whole antibody is divided into sub-antibodies and each sub-antibody represents a sequence of operations on a single machine. Each sub-antibody is combined with cells and each of these cells represents one operation of a certain product part number. Because every operation can only be operated on a certain machine, operations can only swap positions within the same machine; hence mutation operations are performed on cells within the same sub-antibody. The cell and antibody encoding scheme is illustrated in Figure 7-2 and Figure 7-3.

Oper No. = Operation number
Other Info = Other information (e.g. Set up, machining and transfer time, product structure level etc.)

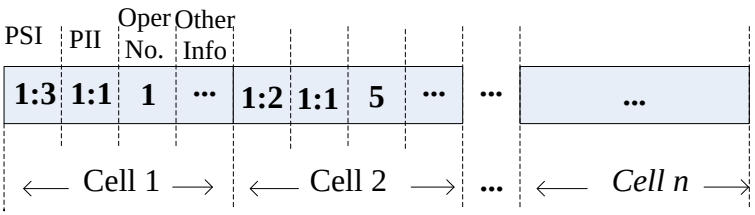

Figure 7-2. Cell representation

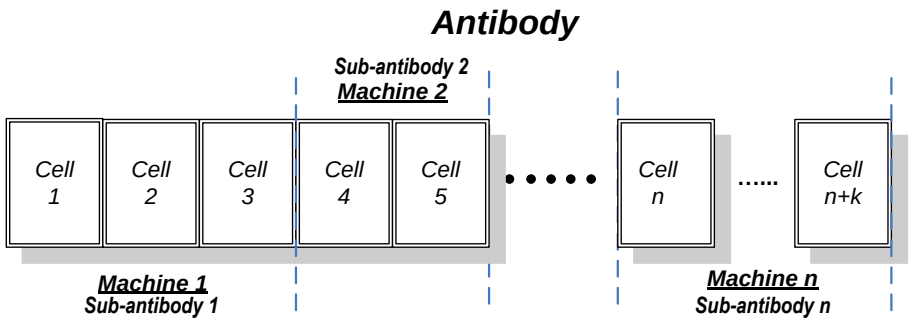

Figure 7-3. Antibody representation

7.2.3 Antibody initialisation and parameter setting

Each of antibodies is linked and the total number of antibodies can be extended to a desire size. The antibody population size (*P*), number of iterations (*I*) and the percentage of antibody elimination (*%B*) are set prior to the program execution.

7.2.4 Repair processes

The repair process rectifies infeasible schedules that contravene operations or assembly precedence relationships, or that are not illegible due to finite capacity constraints. The ESOGAST repair process was used. There are four steps within the repair process: 1) operation precedence adjustment; 2) part precedence adjustment; 3) deadlock adjustment; 4) timing assignment and capacity considerations. Details was described in section 3.2.5

7.2.5 Affinity measurement

The affinity measurement is a procedure that decides whether an antibody will be placed in a new population or replaced by clones. The affinity value is determined by the affinity function. The affinity measurement is conducted on two processes: the cloning selection process and the affinity maturation process. The cloning process decides how many clones are generated proportionate to its antibody affinity value. The affinity maturation consists of two processes: mutation and receptor editing. In mutation, a two phased mutation processes was used. The generated clones undergo an inverse mutation procedure at first and pairwise interchange mutation next. In case there isn't any better solution generated, the procedure keeps the original one (generated clone). The receptor editing process is conducted by eliminating antibodies from the population based upon the percentage of antibody elimination (%B).

The affinity function is devised from the fitness function of ESOGAST and it is described as follows:

$$Total\ cost\ = tardiness\ penalties + earliness\ penalties$$
$$F = P_t \sum_{i=1}^{n} T_{pi} + P_e(\sum_{i=1}^{n} E_{ci} + \sum_{i=1}^{n} E_{ai} + \sum_{i=1}^{n} E_{pi}) \quad \text{Equation 7-1)}$$

Where $T_{pi} = \max(0, F_{pi} - D_{pi})$ $E_{pi} = \max(0, D_{pi} - F_{pi})$

$E_{ci} = \max(0, D_{ci} - F_{ci})$ $E_{ai} = \max(0, D_{ai} - F_{ai})$

Notation: subscript *a, c, p, i* (assembly, component, product, item number), *n* is the number of the according items (either assembly, or component, or product)

$P_t = Penalty\ rate\ of\ tardiness\ (pounds\ perday)$

$P_e = Penalty\ rate\ of\ earliness\ (pounds\ per\ day)$

$T_{pi} = Tardiness\ of\ product\ i\ (days)$ $E_{pi} = Earliness\ of\ product\ i\ (days)$

$E_{ci} = Earliness\ of\ component\ i\ (days)$ $E_{ai} = Earliness\ of\ assembly\ i\ (days)$

$F_{pi} = Finish\ time\ of\ product\ i\ (date)$ $D_{pi} = Due\ date\ of\ product\ i\ (date)$

$F_{ci} = Finish\ time\ of\ component\ i\ (date)$ $D_{ci} = Due\ date\ of\ component\ i\ (date)$

$F_{ai} = Finish\ time\ of\ assembly\ i\ (date)$ $D_{ai} = Due\ date\ of\ assembly\ i\ (date)$

7.3 Specification of the AISST Model

This section provides a functional specification of the AISST and summarises its construction and technical features. A flowchart of AISST is also provided.

7.3.1 General principles and data structure

The same general principles and data structure are applied as described in section 3.3.1 and 3.3.2 of Chapter 3 for ESOGAST.

7.3.2 Input/output files

This AISST have the same input/output data files as described in section 3.3.3 of Chapter 3 for ESOGAST.

7.3.3 Functional specification of the AISST program

The sequences of the AISST procedure are shown as follows:

1) Parameter setting: the program enables executer to input the parameters as desired. These parameters are: problem size, population size (*P*), number of iterations (*I*), and the percentage of antibody elimination (*%B*) (go to 2);

2) Population initialisation: there are two steps to initialise population.
    a) A string of cells (antibody) is generated and stored, which contains the data information from the input text file. This single antibody is the initial antibody to create a population and it uses double-linked structure (go to 2b);
    b) All the cells within its sub-antibody of this initial antibody are non-repeatedly and randomly selected and copied one by one. These cells then are put together to generate a new sub-antibody. After all the new sub-antibodies are generated, a new antibody then is formed (go to 2c);
    c) Step 2b is repeated until the number of antibodies generated equal to the specified population size. The initial population is then produced (go to 3);
3) Repair process: the repair process of the ESOGAST is used by AISST to rectify the infeasible and illegal antibodies (schedules) in the initial population (go to 4);
4) Affinity measurement: the fitness function is used to evaluate the total penalty as the affinity of antibodies. It has the following sequences:
    a) The finish time of the final product is compared with its due date. If final product is finished late, lateness time is calculated. Earlier completion time for every component, part and final product is compared with their due date as well and is accumulated. The penalty cost associated with the schedule can be determined by the lateness and earliness times as described on Equation 7-1 in section 7.2.5 (go to 4b);
    b) Step 4a is repeated for all antibodies (go to 5);
5) Clone: Antibody is now cloned and the number of clones is proportionate with its affinity. The clone process continues until the number of antibody meets the population size. Now the new cloned antibodies form the new population (go to 6);
6) Inverse mutation: each antibody now performs inverse mutation:
    a) Each antibody mutates by applying this mutation operator (go to 6b);
    b) Repeat 6a until the number of antibody equals population size (go to 7);
7) Repair process: the repair processes are repeated to rectify the illegal antibodies (infeasible schedules) created by inverse mutation (go to 8);
8) Affinity judgement: each new antibody produced by inverse mutation is compared with the affinity of the clone. This process continues till all the antibodies are evaluated:
    a) If the affinity of mutated antibody is fitter than its clone, this mutate antibody is copied and replaces its clone (go to 12);

b) If the affinity of mutated antibody is not fitter than its clone, this antibody will mutate again (go to 9);

9) Pairwise interchange mutation: antibodies from 9b now perform the pairwise interchange mutation:
   a) This antibody mutates by applying this mutation operator (go to 9b);
   b) Repeat 9a until all the less fit antibodies perform this mutation (go to 10);
10) Repair process: the repair processes are repeated to rectify the illegal antibodies (infeasible schedules) created by pairwise interchange mutation (go to 11);
11) Affinity judgement: the affinity of the repaired antibody is compared with the affinity of the clone. This process continues till all repaired antibodies from 10) are evaluated:
   a) If the affinity of mutated antibody is fitter than its clone, this mutated antibody is copied and replaces its clone in the initial population (go to 12);
   b) If the affinity of mutated antibody is still not fitter than its clone, this antibody is discarded and initial clone is kept. (go to 12);
12) Antibody elimination:
   a) Calculate the number of the antibodies that will be eliminated. The number equals population size times the percentage of antibody elimination (go to 12b);
   b) Find the antibody with the worst affinity and eliminated from the population (go to 12c);
   c) Repeats 12b until the number of antibody found meets the number of antibodies that will be eliminated (go to 12d);
   d) All the eliminated antibodies will be replaced by newly created antibodies (go to 12e);
   e) All the newly created antibodies perform repair process. A new population is now created (go to 13);
13) Result: the antibody with the minimum penalty is stored. The minimum, mean and maximum penalty of the population is also stored (go to 14);
14) Program termination: termination condition is checked. If the condition is satisfied, go to step 15. Otherwise, next generation is required (go to 4);
15) Displaying the result: two output file are produced (go to 16);
16) The AISST is terminated.

A flowchart of the AISST program is shown in Figure 7-4.

7.3.4 Assumptions within the AISST model

This AISST used the same assumptions described in section 3.3.6 of Chapter 3 for ESOGAST.

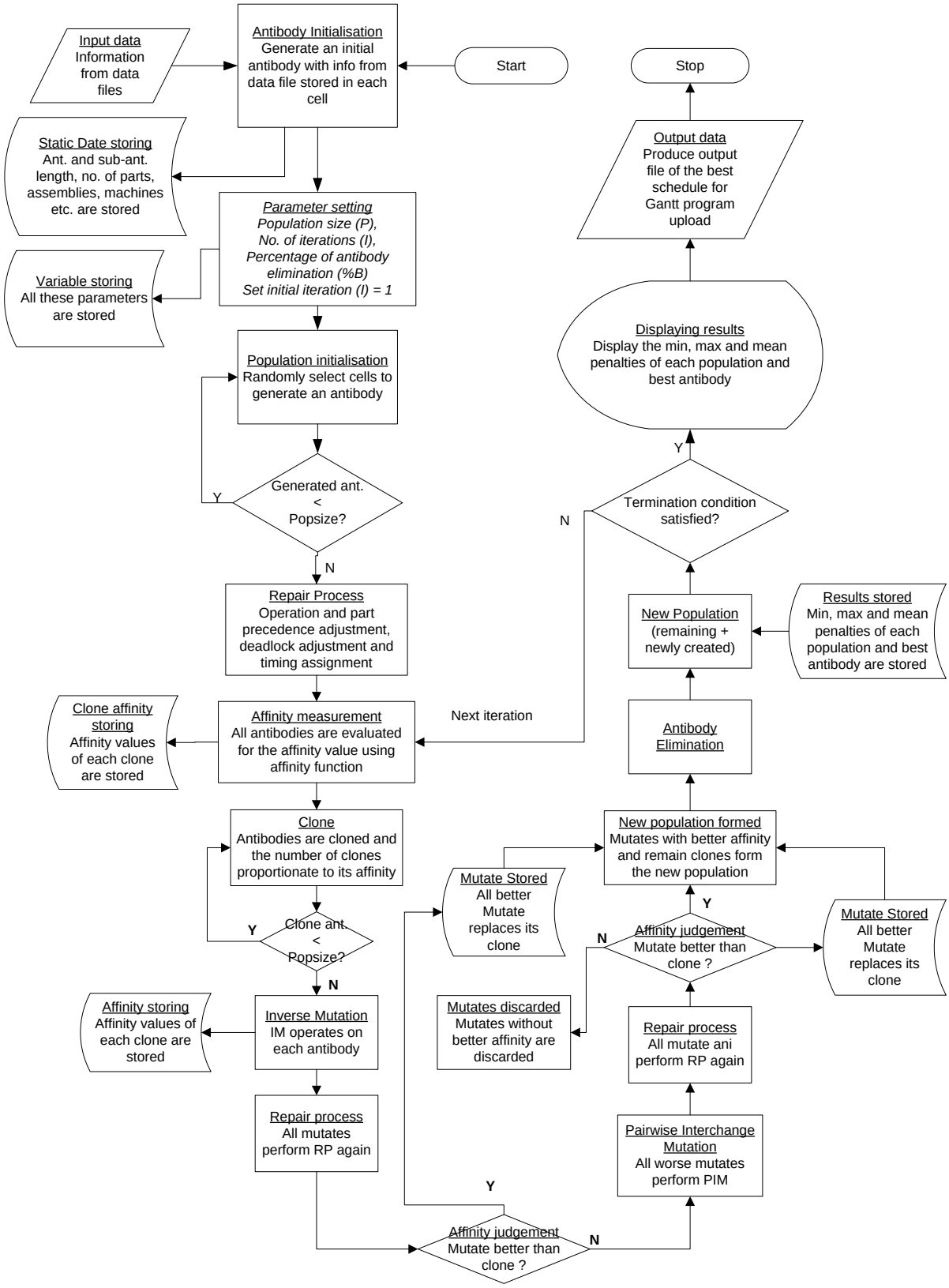

Figure 7-4 Flow chart of AISST program

Chapter 8. Experimental Programme III

This chapter describes the AISST experimental programme, which had the following objectives: 1) to apply AISST to medium, large and extra large schedules and identify the best setting of percentage of antibody elimination; 2) to analyse the results obtained and determine how to choose appropriate parameters by a statistical study of ANOVA; 3) to benchmark and compare the performance between the AISST and the ESOGAST, and also provide analysis of the observed results.

8.1 General Experimental Design

In this experimental programme, four different sizes of industrial production schedules (medium, large, extra large and full size schedule) that shown in Table 4-1 were considered. Note that for the full schedule, assembly time is assumed one month and assembly capacity is assumed as infinite (an assumption adopted by Hicks, 1998). The GA population sizes chosen were 40, 60, and 80 for medium, large, extra large respectively with 20 generations (as used by Pongcharoen (2001)). Population size of 200 with 100 generations was chosen for the full size schedule based on previous experiment in Chapter 4.

8.2 Experiment A

The objective of this experiment was to identify the best setting of percentage of antibody elimination (%B).

8.2.1 Results and discussion

The percentages of antibody elimination tested were 0.05, 0.10, 0.25, 0.50 and 0.75. These values were used by Thapatsuwan et al. (2010). Each run was replicated fifty times with different random seed number for each problem and hence there were 750 run in total in this experiment. Table 8-1 illustrates the results obtained by AISST with different %B. It can be seen that the best %B both in terms of minimum and mean penalty were 0.10, 0.25 and 0.50 for medium, large and extra large schedules respectively with those numbers shown in shadow bold. In terms of the minimum penalties, results were similar with different %B. In terms of mean penalty, there was no significant difference among %B of 0.05, 0.10, 0.25, and 0.5. However the mean penalties increase dramatically with %B of 0.75.

%B	Medium problem		Large problem		Extra large problem	
	Minimum	Mean	Minimum	Mean	Minimum	Mean
0.05	3145.2	3168.9	6129.3	6505.6	296909.3	311387.8
0.10	**3137.5**	**3155.7**	6035.8	6474.1	294050.7	317144.4
0.25	3148.5	3160.1	**5918.3**	**6243.2**	290234.1	312368.1
0.50	3138.3	3174.4	5962.4	6384.1	**285687.1**	**307221.8**
0.75	3142.4	3673.5	6185.8	10371.0	302698.5	515745.7

Table 8-1. Minimum and Mean penalty with different %B

8.3 Experiment B

In this experiment, a further analysis was conducted to compare the mean minimum and mean penalties with different %B in order to determine whether a particular point or a set of range of %B were suitable for AISST.

8.3.1 Results and discussion

The results are shown in Table 8-2, Table 8-3, and Table 8-4. The test groups those results with similar means into subsets. A value of *Sig.* smaller than 0.05 indicates that there is a significant difference among each subset.

Ryan-Einot-Gabriel-Welsch Range Homogeneous Subsets Displayed (Subset for alpha = 0.05)							
	Medium		Large		Extra large		
N	%B	1	%B	1	%B	1	2
50	.10	3137.5	.25	5918.3	.50	285687.1	
50	.50	3138.3	.50	5962.4	.25	290234.1	
50	.75	3142.4	.10	6035.8	.10	294050.7	294050.7
50	.05	3145.2	.05	6129.3	.05	296909.3	296909.3
50	.25	3148.5	.75	6185.8	.75		302698.5
Sig.		.631		.196		.051	.187

Table 8-2. Comparison of means of minimum penalty with different %B

In Table 8-2, the means of minimum penalty were considered. For medium and large schedule, all the means are in the same subset with value of *Sig.* more than 0.05 which indicates that there is no significant difference among means with different %B. For the extra large problem, there are two groups with similar means. Means with %B of 0.05, 0.10, 0.25 and 0.5 were fairly similar in subset 1 and means with %B of 0.05, 0.10 and 0.75 were also similar in subset 2. The biggest difference of means occurred with %B of 0.5 and 0.75. The results suggested that there was no significant advantage on the minimum results using different values of %B for relatively small problem such as

medium and large schedule. However, when problem became relatively larger, there was a likely impact on the results with different %B.

In Table 8-3 in which the means of mean penalty were considered, it can be observed that the best %B for medium, large and extra large problems were 0.10, 0.25 and 0.50 respectively. For all three problems, there were two subset groups with similar means: group 1 that with %B of 0.05, 0.10, 0.25 and 0.50 and group 2 that with %B of 0.75. There was no significant difference in all the means within subset group 1 with *Sig.* much large than 0.05. The results suggested that there was no significant advantage using different values of %B among 0.05, 0.10, 0.25 and 0.50. However, results also showed that there was a significant difference of means using %B 0.75 compared to means with other %B. Hence a further test was conducted to identify it.

Test type: Ryan-Einot-Gabriel-Welsch Range Homogeneous Subsets Displayed (Subset for alpha = 0.05)									
	Medium			Large			Extra large		
N	%B	1	2	%B	1	2	%B	1	2
50	.10	3155.7		.25	6243.2		.50	307221.8	
50	.25	3160.1		.50	6384.1		.05	311387.8	
50	.05	3168.9		.10	6474.1		.25	312368.1	
50	.50	3174.4		.05	6505.6		.10	317144.4	
50	.75		3673.5	.75		10371.0	.75		515745.7
Sig.		.356	1.000		.182	1.000		.107	1.000

Table 8-3. Comparison of means of mean penalty with different %B

In Table 8-4, means value with %B of 0.75 were compared to means with other %B. Results, which is indicated by number of *Sig.* equals zero (*Sig.* << 0.05) showed that for all three problems, the %B value of 0.75 was statistically significant and produced very high penalty costs compared to other values of %B. This suggested that, according to the mean value, %B should set anywhere between 0.05-0.5; a very high value of %B such as 0.75 is not favorable.

Games-Howell test	%B	%B	Mean Difference (I-J)	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
Medium	.75	.05	504.6	.000	465.2	544.0
		.10	517.8	.000	479.7	555.9
		.25	513.3	.000	474.6	552.1
		.50	499.1	.000	461.7	536.6
Large	.75	.05	3865.4	.000	3487.6	4243.2
		.10	3896.9	.000	3536.3	4257.5
		.25	4127.8	.000	3754.8	4500.8
		.50	3986.8	.000	3642.7	4331.0
Extra large	.75	.05	204357.9	.000	192160.8	216555.0
		.10	198601.3	.000	188194.5	209008.0
		.25	203377.6	.000	192225.0	214530.2
		.50	208523.8	.000	198582.3	218465.4

Table 8-4. Means of mean penalty comparison (0.75 vs. 0.05, 0.10, 0.25, 0.50)

8.4 Experiment C

The aim of this experiment was to compare the performance of the AISST and the ESOGAST for medium, large and extra large schedules with the same amount of search in a population or iteration (Antibody size * number of iteration = population size * number of generations (P*I = P*G)). The configuration used for the ESOGAST was crossover probability of 0.8 and mutation probability of 0.1, Partially Mapped Crossover (PMX) (Goldberg and Lingle, 1985), Two Operations Adjacent Swap (2OAS) (Murata and Ishibuchi, 1994) and Rank-based Roulette-Elitist strategy (Forrest et al.) (Tunnukij and Hicks, 2009). The configuration was used based upon experiments conducted in Chapter 4 of this research.

8.4.1 Results and discussion

The best schedules obtained by AISST and ESOGAST were compared in terms of minimum penalty. The results are shown in Table 8-5. For the medium, large schedule, the results clearly shows that AISST performed better than ESOGAST. However, for the extra large schedule, the ESOGAST worked better than the AISST. A more clear comparison is showed in Figure 8-1 for all 50 replicates of each problem. As it can be seen from the figure, as the problem becomes bigger, ESOGAST produced a better schedule than AISST. However, in all of these problems, AISST provided a better mean value and also a smaller variance. The computational time of both tools shown in Table 8-6 indicated that the computation time of AISST was about three to four times longer

than that of ESOGAST. The computation time of AISST varied with different %B and the time increased as the size of %B grows.

	Lowest minimum penalty obtained	
	AISST	ESOGAST
Medium	3124.5	3124.5
Large	4724.0	6368.2
Extra large	250500.3	231237.1

Table 8-5.Best results obtained by AISST and ESOGAST

Problem types / Tools	Medium	Large	Extra large	Full schedule
CPU time (AISST)	<=1s	3~4s	30~44s	≈2.6~3.9*10^4s
CPU time (ESOGAST)	<=1s	≈2s	≈14s	≈1*10^4s

Table 8-6. Computational time of AISST and ESOGAST for each run

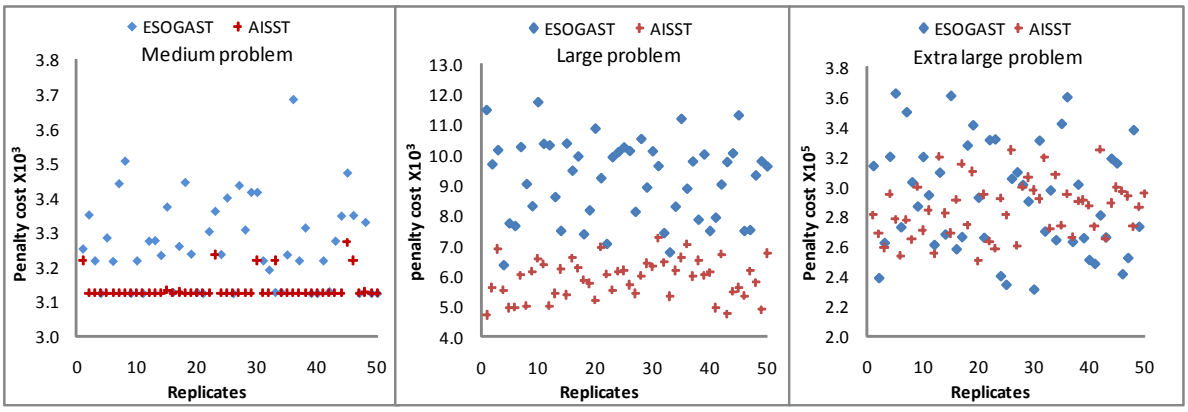

Figure 8-1. Comparison between ESOGAST and AISST

8.5 Experiment D

In this experiment, the AISST was applied for the full size schedule and the results of minimum penalty were compared with that of ESOGAST.

8.5.1 Results

It can be seen in Figure 8-2 that ESOGAST achieved much better result than AISST with all five different %B applied. This coordinates the analysis in section 8.4. It also showed that AISST performed better with relatively smaller %B. For example, AISST with %B of 0.05, 0.10 and 0.25 achieved better results than those with 0.50 and 0.75.

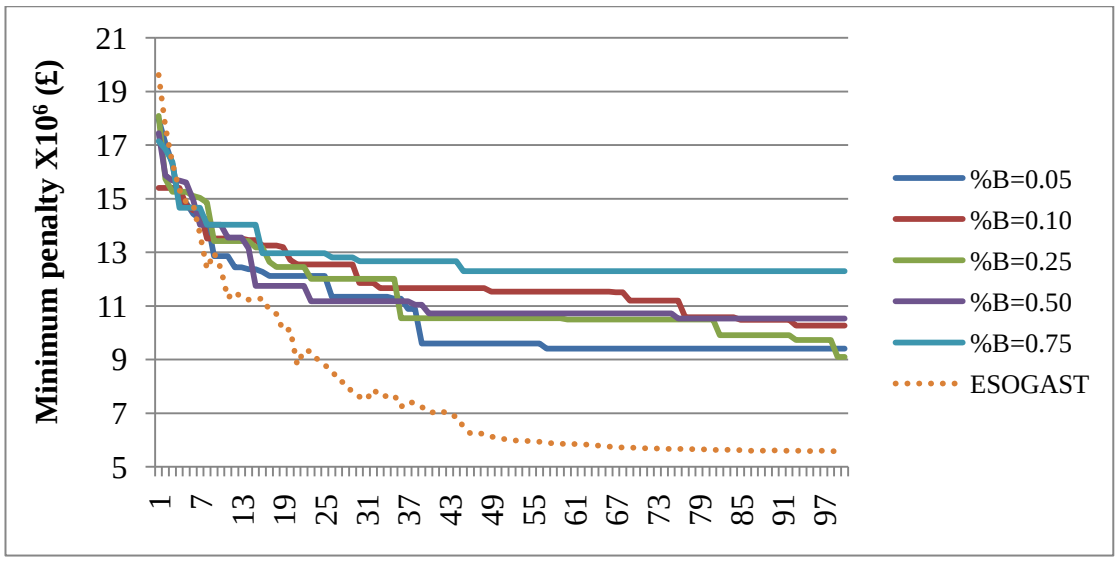

Figure 8-2. Performance comparison (full size schedule)

8.6 Discussion and Conclusions

Production scheduling for Capital Goods Company is very difficult because of the complex product structure and manufacturing process. This work has developed a practical scheduling tool named Artificial Immune System Scheduling Tool (AISST) to tackle such production scheduling problems encountered in such environment. This tool aims to minimise the total penalty cost due to earliness and tardiness completion of components, assemblies and final products. Four experiments were designed and conducted. Results obtained from Experiment A and B suggest that there is no significant impact on the results in terms of minimum penalty using different %B in relatively small problems. However when problems become bigger, this is not necessary the case and the factor of %B becomes more important. In terms of mean penalty, results showed that there is no significant impact on the results with %B less than 0.5. This suggests that %B should be set between 0.05 and 0.5. However, result with %B of 0.75 was very poor. This may suggest that when there is a large percentage of the antibodies in a population that are replaced by newly generated antibodies, the mean value increased and it is because the newly generated antibodies have not been carried out affinity measurement and elimination, thus with relatively lower affinity.

In experiment C and D, results generally showed that AISST perform better than ESOGAST when problems are relatively small. However, ESOGAST is superior to AISST when problems get very large such as the full schedule studied in this paper. Within the AIS algorithm structure, there are three affinity measurement mechanisms to choose the solutions with better affinity value. These mechanisms act as Elitist liked

strategies in GA that select fitter individuals into next generation. This enables AISST to converge faster. So when the search space is relatively small, this may suit AIS more than GA to find results faster and better. However when the search space becomes tremendously large, such as the full schedule with $(1017!)^{36}$ potential solutions, GA may work better than AIS with more exploration and exploitation mechanisms by performing various crossover and mutation operators. The various genetic operators may enable GA to perform a better global search. Compared to the ESOGAST, the AISST only has limited mutation operators with inverse mutation and pairwise interchange mutation and has more convergence encouraged mechanism such as affinity measurements in a single iteration. Furthermore, the mutation rate of AISST will decrease by generations as fitter antibodies are generated; however in ESOGAST, the mutation rate is fixed and would not change throughout generations. This ensures GA more likely to find better individuals in later generations. The computational time of AISST is several times longer than that of ESOGAST. This is because that the AISST has a more complex structure with three affinity measurements and three repair processes in one iteration, while the ESOGAST only has one fitness measurement and one repair process in one generation. Tests have shown that in every single generation, the repair process and fitness measurement occupied over 90% of the computational time of the ESOGA. The computation time of AISST varies with different %B. The bigger the %B, the longer it takes to run.

Chapter 9. Conclusions and Further Work

This chapter presents the contributions and conclusions of this research. Recommendations for further work are also suggested and described.

9.1 Contributions and Conclusions

Products manufactured in capital goods companies have special characteristics compared to products in other manufacturing companies which was described in section 2.1. The production scheduling problems are NP-complete and a practical approach is required for solving complex problems in a realistic time. To produce a schedule with multi-objectives is necessary in these companies. In capital goods industry, products are highly customised and demand is highly variable and uncertain. Capital goods companies aim to achieve best delivery performance to meet customer requirements, whilst minimise inventory costs before final products are delivered to the customers. Companies also need to improve production efficiency by maximising resource utilisation. All of these objectives will help companies to maintain competitiveness and market share and they are crucial for companies to survive. A practical tool that can optimise multiple objectives is required.

The major contributions of this research are described in the following regarding to three scheduling tools that developed in this work:

Enhanced Single-Objective Genetic Algorithm Scheduling Tool (ESOGAST)

1) An Enhanced Single-Objective Genetic Algorithm Scheduling Tool (ESOGAST) to produce optimal schedules to minimise earliness and lateness penalty caused by late or early arrival of components, assemblies and final product for capital goods companies was developed;
2) The ESOGAST was compiled in C and has been tested using company data. The computational time of this tool is up to more than 1300 times faster than the previous tool (GAST) developed by Pongcharoen (2001). This made it possible for ESOGAST to solve very large scheduling problems using full factorial design;
3) Due to the speed of the ESOGAST, it allows the tool to use much large population size with more generations. It also benefited from the development of better selection schemes that were developed in this research. The quality of solutions produced by ESOGAST was compared with GAST. It produced schedules that

reduced minimum penalty cost by 5.3%, 31% and 46% for small, medium and large problems respectively;

4) The performance of the ESOGAST was compared with random search with repair process. Results identified that the repair process within the ESOGAST did not degrade the power of the GA and the ESOGAST was much superior to random search with repair process;
5) The experiments identified the best combination of crossover and mutation operators. Better selection schemes were also developed and identified in this research;
6) The ESOGAST was applied to solve a very large scheduling problem using data from a collaborative company manufactures complex capital goods products. The tool achieved satisfactory results in a realistic time.

Multi-Objective Genetic Algorithm Scheduling Tool (MOGAST)

1) A Multi-Objective Genetic Algorithm Scheduling Tool was develop to minimise inventory cost while simultaneously optimise delivery performance and resource utilisation for production scheduling problems in capital industry companies;
2) A range of crossover and mutation probability of GA were evaluated and the best settings are identified for various industrial scheduling problems;
3) The experiments were conducted to identify the best combination of crossover, mutation operators and selection schemes that were developed in this research. Results were compared to previous study;
4) An evaluation of different niche size (σ) on the performance of the MOGAST was conducted and the best value for various problems were identified;
5) The trade-off or conflicting nature of all three objectives were studied and identified;
6) MOGAST was applied to solve full size schedule problems using the best configuration of MOGAST based on results obtained from previous experiments. The tool achieved satisfactory results in a realistic time.

Artificial Immune System Scheduling Tool (AISST)

1) An Artificial Immune System Scheduling Tool (AISST) was developed to minimise the penalty costs caused by early or late completion of components, assemblies and final products. This is the first tool that applied AIS to solve scheduling problems in capital goods industry;

2) Determine how to choose appropriate percentage of antibody elimination of the AISST by a statistical study of ANOVA;
3) Compared the performance of the AISST with ESOGAST on various scheduling problems in terms of quality of solutions on same amount of search and the computation time required. This experiment benchmarked the performance of both tools and provided comparison studies between these two tools for future work.

9.2 Recommendations for Further Work

Although the computation time of the three tools developed in the research was dramatically reduced, the structures within the C coded program could be improved further; hence the speed of the tool could be reduced. The crossover and mutation operators used in this research were widely applied for travelling salesman problem in the literature. There are newly developed operators that aim to apply for some specific scheduling problems. These operators could be applied and evaluated in the future. Niche size although evaluated in this research, but the value is static in each generation. As the niche size will likely vary in following generation, a dynamic niche size could be applied to better represent the sharing effects between solutions in different generation.

All the tools developed in this research used deterministic data and hence they are deterministic scheduling tools. These tools could be developed into stochastic scheduling tools by add some stochastic processes. This research was based on a static scheduling environment, a more realistic scheduling environment e.g. dynamic scheduling could be considered. Real-time events encountered in dynamic scheduling such as machine breakdown, operator illness, arrival of urgent job, change of due date etc. could be considered.

There is a trend for future study of Metaheuristic Algorithms to combine different nature inspired mechanism to improve their performance. So it could be a good idea to propose a hybrid GA-AIS scheduling tool that combines the advantages of both algorithms to improve the performance.

Appendices

Appendix A – A Timeline of Main Contributions to Metaheuristic

Table A- 1 Main contributions to Metaheuristic in literature

Year	Major contribution in the field
1952	Robbins and Monro work on stochastic optimization methods.(Robbins and Monro, 1951).
1952	Fermi and Metropolis develop an early form of pattern search as described belatedly by Davidon (Davidon, 1991).
1954	Barricelli carry out the first simulations of the evolution process and use them on general optimization problems (Barricelli, 1954).
1963	Rastrigin proposes random search (Rastrigin, 1963).
1965	Matyas proposes random optimization (Matyas, 1965)
1965	Rechenberg proposes evolution strategies (Rechenberg, 1965)
1965	Nelder and Mead propose the Simplex method (Nelder and Mead, 1965)
1966	Fogel et al. propose evolutionary programming (Fogel et al., 1966)
1970	Hastings proposes the Metropolis-Hastings algorithm (Hastings, 1970)
1970	Cavicchio proposes adaptation of control parameters for an optimizer (Cavicchio, 1970)
1975	Holland proposes the genetic algorithm (Holland, 1975)
1978	Mercer and Sampson propose a metaplan for tuning an optimizer's parameters by using another optimizer (Mercer and Sampson, 1978)
1980	Smith describes genetic programming (Smith, 1980)
1983	Kirkpatrick et al. propose simulated annealing (Kirkpatrick et al., 1983)
1986	Glover proposes tabu search, first mention of the term *metaheuristic* (Glover, 1986)
1986	Farmer et al. work on the artificial immune system (Farmer et al., 1986)
1986	Grefenstette proposes another early form of metaplan for tuning an optimizer's parameters by using another optimizer (1986)

Table A- 1 Continued

Year	Major contribution in the field
1988	First conference on genetic algorithms is organized at the University of Illinois at Urbana-Champaign
1988	Koza registers his first patent on genetic programming (Koza, 1988)
1989	Goldberg publishes a well known book on genetic algorithms (Goldberg, 1989)
1989	Evolver, the first optimization software using the genetic algorithm
1989	Moscato proposes the memetic algorithm (Moscato, 1989)
1991	Interactive evolutionary computation
1992	Dorigo proposes the ant colony algorithm (Dorigo, 1992)
1993	The journal, *Evolutionary Computation*, begins publication by the Massachusetts Institute of Technology.
1993	Fonseca and Fleming propose MOGA for multiobjective optimization (Fonseca and Fleming, 1993)
1994	Srinivas and Deb propose NSGA for multiobjective optimization (Srinivas and Deb, 1994)
1995	Kennedy and Eberhart propose particle swarm optimization (Kennedy and Eberhart, 1995)
1995	Wolpert and Macready prove the no free lunch theorems (Wolpert and Macready, 1995, Wolpert and Macready, 1997)
1996	Mühlenbein and Paaß work on the estimation of distribution algorithm (Mülhenbein and Paaß, 1996)
1996	Hansen and Ostermeier propose CMA-ES (Hansen and Ostermeier, 1996)
1997	Storn and Price propose differential evolution (Storn and Price, 1997)
1997	Rubinstein proposes the cross entropy method (Rubinstein, 1997)
2001	Geem et al. propose harmony search (Geem et al., 2001)
2002	Deb et al. propose NSGA-II for multiobjective optimization (Deb et al., 2002)
2004	Nakrani and Tovey propose bee colony optimization (Nakrani and Tovey, 2004)
2005	Krishnanand and Ghose propose Glowworm swarm optimization (Krishnanand and Ghose, 2005)
2005	Karaboga proposes Artificial Bee Colony Algorithm (ABC) (Karaboga, 2005)
2006	Haddad et al. introduces honey-bee mating optimization (Haddad, 2006)

Table A- 1 Continued

Year	Major contribution in the field
2007	Hamed Shah-Hosseini introduces Intelligent Water Drops. (Shah-Hosseini, 2009)
2008	Yang introduces firefly algorithm (Yang, 2008)
2008	Mucherino and Seref propose the Monkey Search (Mucherino and Seref, 2008)
2009	Yang and Deb introduce cuckoo search (Yang and Deb, 2009)
2010	Pedersen proposes tuning and simplifying optimizers (Pedersen, 2010)

All resources obtained from (Wikipedia, 2010)

Appendix B – A Classification of Multi Objective Optimisation Algorithms and A List of Description of Popular Multi-Objective Genetic Algorithms

Table B - 1 A classification of Multi-Objective Optimisation Algorithms (Deb, 2004)

Classical Methods	Goal Programming Method (Charnes et al., 1955);	ε-Constraint Method (Haimes et al., 1971) ;	Benson's Method (Benson, 1978)
	Minimum Deviation Method (Tabucanon, 1989);	Single Objective Approach (Tabucanon, 1989);	Global Criterion Method (Tabucanon, 1989);
	Utility Function Method (Tabucanon, 1989);	Weighted Sum Method (Syswerda and Palmucci, 1991);	Goal Attainment (Wilson and Macleod, 1993);
	Weighted Metric Method (Miettinen, 1999);	Value Function Method (Miettinen, 1999);	Interactive Methods (Deb, 2004)
Non-Elitist MOEAs	Vector Evaluated Genetic Algorithm (Schaffer, 1985);	Non-Dominated Sorting GA (Srinivas and Deb, 1994);	
	Vector-Optimized Evolution Strategy (Kursawe, 1990);	Weight-Based Genetic Algorithm (Hajela et al., 1993);	
	Multi-Objective GA (Fonseca and Fleming, 1993);	Niched-Pareto GA (Horn et al., 1994);	Random Weighted GA (Murata and Ishibuchi, 1995);
	Neighbourhood Constrained GA (Loughlin and Ranjithan, 1997);	Modified NESSY GA (Koppen et al., 1997);	
	Predator-Prey Evolution Strategy (Laumanns et al., 1998);	Multi-objective Distributed Q-learning Algorithm (Romero and Manzanares, 1999);	
	Distributed Sharing GA (Hiroyasu et al., 1999);	Multi-Objective Ant-Q Algorithm (Romero and Manzanares, 1999);	
	Pareto Envelope-based Selection Algorithm (Corne et al., 2000);	Distributed Reinforcement Learning Approach (Mariano and Morales, 2000);	
	Nash GA (Sefrioui and Periaux, 2000);	Clustering Pareto Evolutionary Algorithm (Molyneaux et al., 2001);	
	Multi-objective Immune System Algorithm (Coello and Cortes, 2002);	Dynamic Multi-Objective Evolutionary Algorithm (Lu and Yen, 2002)	
	Mendelian Multi-Objective Simple Genetic Algorithm (Kadrovach et al., 2002);	Multi-Objective Immune Algorithm (Luh et al., 2003)	

Table B - 1 Continued

Elitist MOEAs	Multi-Objective Micro-GA (Krishnakumar, 1989); Distance-Based Pareto GA (Osyczka and Kundu, 1995); Thermo-dynamical GA (Kita et al., 1996);	
	Non-Dominated Sorting in Annealing GA (Leung et al., 1998);	Strength Pareto EA (Zitzler and Thiele, 1998);
	Multi-Objective Messy GA (Veldhuizen, 1999);	Multi-Objective Evolutionary Algorithm (Tan et al., 1999);
	Pareto-Archived Evolution Strategy (Knowles and Corne, 2000);	Strength Pareto Evolutionary Algorithm (Zitzler and Thiele, 1998);
	Elitist MOEA with Co-evolutionary Sharing (Neef et al., 1999);	Rudolph's Elitist MOEA (Rudolph, 2001);
	Pareto Converging GA (Kumar and Rockett, (in press));	Micro-Genetic Algorithm (μGA) (Coello and Pulido, 2001);
	Elitist Non- Dominated Sorting GA (Deb et al., 2002);	Multi-objective Bayesian Optimisation Algorithm (Khan et al., 2002)
	Culture Algorithm with Evolutionary Programming (Coello and Becerra, 2003)	
Constrained MOEAs	Jimenez-Verdegay-Gomez-Skarmeta's Method (Jimenez et al., 1999) ;	Ray-Tai-Seow's Method (Ray et al., 2001)
	Constrained Tournament Method (Deb, 2004);	Penalty Function Approach (Deb, 2004)

Table B - 2. A list of descriptions of popular multi-objective genetic algorithms (Konak et al., 2006)

Algorithms (Author, year)	**Fitness assignment**	**Diversity mechanism**	**Elitism**	**External population**	**Advantages**	**Disadvantages**
VEGA (Schaffer, 1985)	Each subpopulation is evaluated with respect to a different objective	No	No	No	First MOGA; Straightforward implementation	Tend converge to the extreme of each objective
MOGA (Fonseca and Fleming, 1993)	Pareto ranking	Fitness sharing by niching	No	No	Simple extension of single objective GA	Usually slow convergence; Problems related to niche size parameter
WBGA (Hajela et al., 1993)	Weighted average of normalized objectives	Niching Predefined weights	No	No	Simple extension of single objective GA	Difficulties in nonconvex objective function space
NPGA (Horn et al., 1994)	No fitness assignment tournament selection	Niche count as tiebreaker in tournament selection	No	No	Very simple selection process with tournament selection	Problems related to niche size parameter; Extra parameter for tournament selection
RWGA (Murata and Ishibuchi, 1995)	Weighted average of normalized objectives	Randomly assigned weights	Yes	Yes	Efficient and easy implement	Difficulties in nonconvex objective function space;
PESA (Corne et al., 2000)	No fitness assignment	Cell-based density	Pure elitist	Yes	Easy to implement; Computationally efficient	Performance depends on cell sizes; Prior information needed about objective space
PAES (Knowles and Corne, 2000)	Pareto dominance is used to replace a parent if offspring dominates	Cell-based density as tie breaker between offspring and parent	Yes	Yes	Random mutation hill-climbing strategy; Easy to implement; Computationally efficient	Not a population based approach; Performance depends on cell sizes

Table B - 2. Continued

Algorithms	Fitness assignment	Diversity mechanism	Elitism	External population	Advantages	Disadvantages
NSGA (Srinivas and Deb, 1994)	Ranking based on non-domination sorting	Fitness sharing by niching	No	No	Fast convergence	Problems related to niche size parameter
NSGA-II (Deb et al., 2002)	Ranking based on non-domination sorting	Crowding distance	Yes	No	Single parameter (N) Well tested; Efficient	Crowding distance works in objective space only
SPEA (Zitzler and Thiele, 1998)	Raking based on the external archive of non-dominated solutions	Clustering to truncate external population	Yes	Yes	Well tested; No parameter for clustering	Complex clustering algorithm
SPEA-2 (Zitzler et al., 2001)	Strength of dominators	Density based on the k-th nearest neighbour	Yes	Yes	Improved SPEA; Make sure extreme points are preserved	Computationally expensive fitness and density calculation
RDGA (Lu and Yen, 2003)	The problem reduced to bi-objective problem with solution rank and density as objectives	Forbidden region cell-based density	Yes	Yes	Dynamic cell update Robust with respect to the number of objectives	More difficult to implement than others
DMOEA (Lu and Yen, 2002)	Cell-based ranking	Adaptive cell-based Density	Yes (implicitly)	No	Includes efficient techniques to update cell Densities; Adaptive approaches to set GA parameters	More difficult to implement than others

Appendix C – The Processes of Artificial Immune System Inspired Algorithms

Table C - 1 Process of negative selection algorithm

```
input   : S_seen = set of seen known self elements
output  : D = set of generated detectors

begin

        repeat
                Randomly generate potential detectors and place them in a set P
                Determine the affinity of each member of P with each member of the self set S_seen
                If at least one element in S recognises a detector in P according to a recognition threshold,
                then the detector is rejected, otherwise it is added to the set of avaliable detectors D
        until   Stopping criteria has been met

end
```

Table C - 2 Algorithm process of CLONALG

```
input   : S = set of patterns to be recognised, n the number of worst elements to select for removal
output  : M = set of memory detectors capable of classifying unseen patterns
begin
        Create an initial random set of antibodies, A
        Forall  patterns in S   do
                Determine the affinity with each antibody in A
                Generate clones of a subset of the antibodies in A with the highest affinity.
                The number of clones for an antibody is proportional to its affinity
                Mutate attributes of these clones to the set A , and place a copy of the highest
                affinity antibodies in A  into the memory set, M
                Replace the n lowest affinity antibodies in A with new randomly generated antibodies
        end
end
```

Table C - 3 Processes of immune network algorithm

```
input   : S = set of patterns to be recognised, nt  network affinity threshold,
          ct  clonal pool threshold, h  number of highest affinity clones, a  number of
          new antibodies to introduce
output  : N = set of memory detectors capable of classifying unseen patterns

begin
        Create an initial random set of network antibodies, N
        repeat
                forall  patterns in  S do
                        Determine the affinity with each antibody in N
                        Generate clones of a subset of the antibodies in N  with the highest affinity. The number of
                                clones  for an antibody is proportional to its affinity
                        Mutate attributes of these clones to the set A , a and place h  number of the highest
                                affinity clones into a clonal memory set, C
                        Eliminate all elements of C  whose affinity with the antigen is less than a predefined
                                threshold ct
                        Determine the affinity amongst all the antibodies in C  and eliminate those antibodies whose
                                affinity with each other is less than the threshold ct
                        Incorporate the remaining clones of C  into N
                end
                Determine the affinity between each pair of antibodies in N  and eliminate all antibodies whose
                        affinity is less than the threshold nt
                Introduce a random number of randomly generated antibodies and place into N
        end until a stopping criteria has been met
end
```

Table C - 4 Processes of Dentritic Cell Algorithm

```
input   : S  = set of data items to be labelled safe or dangerous
output  : D = set of data items labelled safe or dangerous

begin
Create an initial population of dendritic cells (DCs), D
Create a set to contain migrated DCs, M
   forall  data items in S do
   Create a set of DCs randomly selected from D , P
      forall  DCs in P do
         Add data item to DCs collected list
         Update danger, PAMP and safe signal concentrations
         Update concentrations of output cytokines
         Migrate the DC from D   to M  and create a new DC in D  if concentration of co-stimulatory
            molecules is above a threshold
      end
   end

   forall  DCs in M   do
      Set DC to be semi-mature if output concentration of semi-mature cytokines is greater than mature cytokines,
         otherwise set as mature
   end

   forall data items in S   do
      Calculate number of times data item is presented by a mature DC and a semi-mature DC
      Label data item a safe if presented by more than semi-mature DCs than mature DC's,
         otherwise label as dangerous
      Add data item to labelled set M
    end
end
```

All resources obtained from (AISWeb, 2010)

References

Adra, S. (2003) *Optimisation Techniques for Gas Turbine Engine Control Systems.* thesis. Department of Computer Science, University of Sheffield, U.K.

AISWeb (2010) *The Online Home of Artificial Immune Systems*. Available at: http://www.artificial-immune-systems.org/algorithms.shtml (Accessed: 18th August 2010).

Akgunduz, O. S. and Tunal, S. (2010) 'An adaptive genetic algorithm approach for the mixed-model assembly line sequencing problem', *International Journal of Production Research,* 48, (17), pp. 5157-5179.

Al-Hakim, L. (2001) 'An analogue genetic algorithms for solving job shop scheduling problems', *International Journal of Production Research,,* 39, pp. 1537-1548.

Aytug, H., Khouja, M. and Vergara, F. E. (2003) 'Use of genetic algorithms to solve production and operations management problems: a review', *International Journal of Production Research,* v41, (17), pp. 3955-4009.

Bagchi, T. P. (1999) *Multiobjective scheduling by genetic algorithms.* Kluwer Academic Publishers: Boston.

Baker, K. R. (1974) *Introduction to Sequencing and Scheduling.* Wiley and Sons: New York.

Baker, K. R. and Trietsch, D. (2009) *Principles of Sequencing and Scheduling* John & Wiley sons, Inc: Hoboken, New Jersey.

Barricelli, N. A. (1954) 'Esempi numerici di processi di evoluzione', *Methodos,* pp. 45-68.

Benson, H. P. (1978) 'Existence of efficient solutions for vector maximization problems', *Journal of Optimization Theory and Applications,* 26, (4), pp. 569-580.

Blazewicz, J., Domschke, W. and Pesch, E. (1996) 'The job shop scheduling problem: Conventional and new solution techniques', *European Journal of Operational Research,* 93, (1), pp. 1-33.

Brundenius, C. (1987) 'Development and prospects of capital goods production in revolutionary Cuba', *World Development,* 15, (1), pp. 95-112.

Burdett, R. L. and Kozan, E. (2000) 'Evolutionary algorithms for flowshop sequencing with non-unique jobs', *International Transactions in Operational Research,* 7, pp. 401-418.

Burnet, F. M. (1959) *The Clonal Selection Theory of Acquired Immunity.* Cambridge University Press: Cambridge, UK

Cao, P. B. and Xiao, R. B. (2007) 'Assembly planning using a novel immune approach', *International Journal of Advanced Manufacturing Technology,* 31, (7-8), pp. 770-782.

Cavicchio, D. J. (1970) *Adaptive search using simulated evolution.* Technical Report, University of Michigan, Computer and Communication Sciences Department

Celano, G., Fichera, S., Grasso, V., La Commare, U. and Perrone, G. (1999) 'An evolutionary approach to multi-objective scheduling of mixed model assembly lines', *Computers & Industrial Engineering,* 37, (1-2), pp. 69-73.

Chandrasekaran, M., Asokan, P., Kumanan, S., Balamurugan, T. and Nickolas, S. (2006) 'Solving job shop scheduling problems using artificial immune system', *International Journal of Advanced Manufacturing Technology,* 31, (5-6), pp. 580-593.

Chang, C. C., Tseng, H. E. and Meng, L. P. (2009) 'Artificial immune systems for assembly sequence planning exploration', *Engineering Applications of Artificial Intelligence,* 22, (8), pp. 1218-1232.

Charnes, A., Cooper, W. and Ferguson, R. (1955) 'Optimal estimation of executive compensation by linear programming', *Management Science* 1, (2), pp. 138-151.

Chen, J.-S., Pan, J. C.-H. and Wu, C.-K. (2008) 'Hybrid tabu search for re-entrant permutation flow-shop scheduling problem', *Expert Systems with Applications,* 34, (3), pp. 1924-1930.

Choobineh, F. F., Mohebbi, E. and Khoo, H. (2006) 'A multi-objective tabu search for a single-machine scheduling problem with sequence-dependent setup times', *European Journal of Operational Research,* 175, (1), pp. 318-337.

Cochran, J. K., Horng, S.-M. and Fowler, J. W. (2003) 'A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines', *Computers & Operations Research,* 30, (7), pp. 1087-1102.

Coello, C. A. C. (2000) 'An updated survey of GA-based multiobjective optimization techniques ', *ACM Comput. Surv.,* 32, (2), pp. 109-143.

Coello, C. A. C. and Becerra, R. L. (2003) 'Evolutionary multiobjective optimization using a cultural algorithm', *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE.* 6-13

Coello, C. A. C. and Cortes, N. C. (2002) 'An approach to solve multi objective-optimization problems based on an artificial immune system', *First International Conference on Artificial Immune Systems (ICARIS'2002).* University of Kent at Canterbury, Inglaterra, 212-221

Coello, C. A. C. and Pulido, G. T. (2001) 'A micro-genetic algorithm for multi-objective optimisation', *First International Conference on Evolutionary Multi-Criterion Optimization, Lecture Notes in computer Science.*Springer-Verlag.pp 126-140

Coello, C. A. C., Rivera, D. C. and Cortes, N. C. (2003) 'Use of an artificial immune system for job shop scheduling', *Artificial Immune Systems, Proceedings,* 2787, pp. 1-10.

Coello, C. A. C., Rivera, D. C. and Cortes, N. C. (2004) 'Job shop scheduling using the clonal selection principle', *Adaptive Computing in Design and Manufacture Vi,* pp. 113-124.

Coello, C. A. C., Van Veldhuizen, D. A. and Lamont, G. B. (2002) *Evolutionary algorithms for solving multi-objective problems.* Kluwer Academic: New York.

Conway, R. W. (1967) *Theory of Scheduling.* Addison-Wesley: Reading.

Corne, D. W., Knowles, J. D. and Oates, M. J. (2000) 'The Pareto envelope-based selection algorithm for multi-objective optimizaion', *Proceeding of the Parallel Problem Solving from Nature VI Conference.*Springer.pp 839-848

Darwin, C. (1859) *On the origin of species by means of natural selection, or, The preservation of favoured races in the struggle for life.* J.Murray: London.

Davidon, W. C. (1991) 'Variable metric method for minimization', *SIAM Journal on Optimization,* 1, (1), pp. 1-17.

Davis, L. (1985) 'Job shop scheduling with genetic algorithms', *Proceedings of the First Internatinal Conference on Genetic Algorithms and Their Applications.* 136-140

De Jong, K. A. (1975) *An analysis of the behavior of a class of genetic adaptive systems.* Doctoral thesis. University of Michigan,Ann Arbor, MI, .

Deb, K. (2004) *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons, LTD: Chichester.

Deb, K. and Goldberg, D. E. (1989) 'An Investigation of Niche and Species Formation in Genetic Function Optimization', *Proceeding of the third Internatiional Conference on Genetic Algorithm.* Morgan-Kaufmann, 97-106

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002) 'A fast and elitist multiobjective genetic algorithm: NSGA-II', *IEEE Transactions on Evolutionary Computation,* 6, (2), pp. 182-197.

Dimopoulos, C. and Zalzala, M. S. (2000) 'Recent Developments in Evolutionary Computation for Manufacturing Optimization: Problems, Solutions, and Comparisons.', *IEEE Trans.Evol.Comput.*, vol.4, (No.2), pp. pp93-113.

Dorigo, M. (1992) *Optimization, Learning and Natural Algorithms.* PhD thesis. Politecnico di Milano, Italie.

Dorigo, M., Maniezzo, V. and Colorni, A. (1996) 'The ant system: Optimization by a colony of cooperating agents.', *IEEE Transactions on Systems,Man and Cybernetics*, (Part-B 26), pp. 29-41.

Eberhart, R. and Kennedy, J. (1995) 'A new optimizer using particle swarm theory', *Proceeding of the Sixth International Symposium on Micro Machine and Human Science.* Nagoya, Japan, 39–43

Eksioglu, B., Eksioglu, S. D. and Jain, P. (2008) 'A tabu search algorithm for the flowshop scheduling problem with changing neighborhoods', *Computers & Industrial Engineering,* 54, (1), pp. 1-11.

Engin, O. and Döyen, A. (2004) 'A new approach to solve hybrid flow shop scheduling problems by artificial immune system', *Future Generation Computer Systems,* 20, (6), pp. 1083-1095.

Evans, G. W. (1984) 'An overview of techniques for solving multiobjective mathematical programs', *Management Science,* 30, (11), pp. 1268-1282.

Farmer, J. D., Packard, N. and Perelson, A. (1986) 'The immune system, adaptation and machine learning', *Physica,* D, (22), pp. 187-204.

Fogel, L., Owens, A. J. and Walsh, M. J. (1966) *Artificial Intelligence through Simulated Evolution.* Wiley.

Fonseca, C. M. and Fleming, P. J. (1993) 'Genetic algorithms for multiobjective optimization: formulation, discussion and generalization', *Proceedings of the Fifth International Conference on Genetic Algorithms.* San Mateo, CA:, July, 1993.Morgan Kaufmann.pp 416-423

Fonseca, C. M. and Fleming, P. J. (1995) 'An overview of evolutionary algorithms in multiobjective optimization', *IEEE Transactions on Evolutionary Computation,* 3, (1), pp. 1-16.

Fonseca, C. M. and Fleming, P. J. (1998) 'Multiobjective optimization and multiple constraint handling with evolutionary algorithms - Part 1: A unified formulation', *IEEE Transactions on Systems, Man and Cybernetics,,* vol.28, (1), pp. 26-37.

Fry, T. D., Oliff, M. D., Minor, E. D. and Leong, G. K. (1989) 'The effects of product structure and sequencing rule on assembly performance', *International Journal of Production Research,* 27, (4), pp. 671-686.

Ge, H. W., Du, W. L., Feng, Q. and Lu, W. (2007) 'An intelligent hybrid algorithm for job-shop scheduling based on particle swarm optimization and artificial immune system', *Analysis and Design of Intelligent Systems Using Soft Computing Techniques,* 41, pp. 628-637.

Ge, H. W., Sun, L. and Liang, Y. C. (2005) 'Solving job-shop scheduling problems by a novel artificial immune system', *Ai 2005: Advances in Artificial Intelligence,* 3809, pp. 839-842.

Geem, Z. W., Kim, J. H. and Loganathan, G. V. (2001) 'A new heuristic optimization algorithm: harmony search', *Simulation,* 76.

Gen, M. and Cheng, R. (1997) *Genetic Algorithms and Engineering Design.* John Wiley and Sons: New York, USA.

Glover, F. (1986) 'Future Paths for Integer Programming and Links to Artificial Intelligence', *Computers and Operations Research,* 13, (5), pp. 533-549.

Glover, F. (1989) 'Tabu Search - Part I', *ORSA Journal on Computing,* 1(3),, pp. 190-206.

Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-wesley,: Reading, MA.

Goldberg, D. E. and Lingle, R. (1985) 'alleles, Loci and the TSP', *Proceedings of the first International Conference on Genetic Algorithms and Their Applications.* Hillsdale, New Jersey:Lawrence Erlbaum.pp 154-159

Goldberg, D. E. and Richardson, J. (1987) 'Genetic Algorithms with Sharing for Multimodal Function Optimization', *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms.* 41-49

Gorczyca, M., Duenas, A. and Petrovic, D. (2004) 'A new multi-objective genetic algorithm for job shop scheduling with limited resources', Wroclaw, Poland:Technical University of Wroclaw, Wroclaw, 50-370, Poland.pp 201-211

Gottipolu, R. B. and Ghosh, K. (2003) 'A simplified and efficient representation for evaluation and selection of assembly sequences', *Computers in Industry,* 50, (3), pp. 251-264.

Greenberg, H. H. (1968) 'A branch and bound solution to the general scheduling problem', *Operation Research,* 16, pp. 353-361.

Greensmith, J. and Aickelin, U. (2007) 'Dendritic ceels for syn scan detection', *In proceeding of the Genetic and Evolutionary Computation conference (GECCO 2007).* 49-56

Greensmith, J., Whitbrook, A. and Aickelin, U. (2010) 'Artificial Immume System', in *Handbook of Metaheuristics.* Springer, pp. tba.

Grefenstette, J. J. (1986) 'Optimization of control parameters for genetic algorithms', *IEEE Transaction on Systems, Man, and Cybernetics,* 16, (1), pp. 122-128.

Haddad, O. B. (2006) 'Honey-bees mating optimization (HBMO) algorithm: a new heuristic approach for water resources optimization', *Water Resources Management,* 20, (661-680).

Haimes, Y. Y., Lasdon, L. S. and Wismer, D. A. (1971) 'On a bicriterion formulation of the problems of integrated system identification and system optimisation', *IEEE Transactions on Systems, Man, and Cybernetics,* 1, (3), pp. 296-297.

Hajela, P., Lee, E. and Lin, C. Y. (1993) 'Genetic algorithms in structural topology optimization.', *In Proceedings of the NATO Advanced Research Workshop on Topology Design of Structures.* 117-133

Hansen and Ostermeier (1996) 'Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation', *Proceedings of the IEEE International Conference on Evolutionary Computation.* 312-317

Hart, E. and Timmis, J. (2008) 'Application areas of AIS: The past, the present and the future', *Applied Soft Computing,* 8, (1), pp. 191-201.

Hastings, W. K. (1970) 'Monte Carlo Sampling Methods Using Markov Chains and Their Applications', *Biometrika,* 57, (1), pp. 97-109.

Held, M. and Karp, R. M. (1962) 'A dynamic programming approach to sewuencing problems', *Journal of Society for Industrial and Applied Mathematics,* 10, pp. 196-210.

Hicks, C. (1998) *Computer Aided Production Management (CAPM) Systems in Make-to-order/Engineer-to-order Heavy Engineering Companies.* Ph.D thesis. University of Newcastle upon Tyne.

Hicks, C. and Braiden, P. M. (2000) 'Computer-aided production management issues in the engineering-to-order production of complex capital goods explored using a simulation approach', *International Journal of Production Research,,* 38, (18), pp. 4783-4810.

Hicks, C. and Earl, C. F. (2001) 'The Validation of Simulation Models', *Assembly Automation, the International Journal of Assembly Technology and Management,* 21, (3), pp. 193-194.

Hicks, C., Earl, C. F. and McGovern, T. (2000) 'An Analysis of Company Structure and Business Processes in the Capital Goods Industry in the UK.', *IEEE Transactions on Engineering Management,* 47, (4), pp. 414-423.

Hicks, C., Song, D. P. and Earl, C. F. (2007) 'Dynamic scheduling for complex engineer-to-order products', *International Journal of Production Research,* 45, (15), pp. 3477-3503.

Hines, S. (1996) *The Development of Graphical User Interface for MTO Simulation Model.* Unpublished paper, Department of Mechanical, Material and Manufacturing Engineering, University of Newcastle upon Tyne, U.K., pp:

Hiroyasu, T., Miki, M. and Watanabe, S. (1999) 'Distributed genetic algorithms with a new sharing approach in multiobjective optimization problems', *In proceeding of the Congress on Evolutionary Computation (CEC-1999).* 69-76

Holland, J. (1975) *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* University of Michigan Press: Ann Arbor.

Horn, J., Nafpliotis, N. and Goldberg, D. E. (1994) 'Niched Pareto genetic algorithm for multiobjective optimization', Orlando, FL, USA:IEEE, Piscataway, NJ, USA.pp 82-87

Hornby, A., S. (2005) *Oxford Advanced Learner's Dictionary.* Oxford University Press.

Islier, A. A. (1998) 'A genetic algorithm approach for multiple criteria facility layout design', *international Journal of Production Research*, 36, pp. 1549-1569.

Jain, A. S. and Meeran, S. (1999) 'Deterministic job shop scheduling: part, present and future', *European Journal of Operation Research*, (113), pp. 390-434.

Jerne, N. (1974) 'Towards a network theory of the immune system', *Ann. Immunology (Inst. Pasteur)*, 125, (C), pp. 373-389.

Jiang, Z., Xuanyuan, S., Li, L. and Li, Z. (2011) 'Inventory-shortage driven optimisation for product configuration variation', *International Journal of Production Research*, 49, (4), pp. 1045-1060.

Jimenez, F., Verdegay, J. L. and Gomez-Skarmeta, A. F. (1999) 'Evolutionary techniques for constrained multiple-objective optimization problems.', *In proceedings of the workshop on Multi-Criterion Optimization Using Evolutionary Methods held at Genetic and Evolutionary Computation Conference (GECCO).* 115-116

Kadrovach, B. A., Zydallis, J. B. and Lamont, G. B. (2002) 'Use of Mendelian pressure in a multi-objevtive genetic algorithm', *Proceeding of the 2002 Congress on Evolutionary Computation.* Honolulu, HI, USA, 962-967

Kahraman, C., Engin, O. and Yilmaz, M. K. (2008) 'A new artificial immune system algorithm for multi objective fuzzy flow shop scheduling: A real world

application', *Computational Intelligence in Decision and Control*, 1, pp. 1087-1092.

Kahraman, C., Engin, O. and Yilmaz, M. K. (2009) 'A New Artificial Immune System Algorithm for Multiobjective Fuzzy Flow Shop Problems', *International Journal of Computational Intelligence Systems*, 2, (3), pp. 236-247.

Karaboga, D. (2005) *An Idea Based On Honey Bee Swarm For Numerical Numerical Optimization.* Technical Report-TR06 (Erciyes University, Engineering Faculty, Computer Engineering Department)

Kennedy, J. and Eberhart, R. (1995) 'Particle Swarm Optimization', *Proceedings of IEEE International Conference on Neural Networks.* 1942-1948

Kernighan, B. W. and Ritchie, D. M. (1988) *The C programming Language.* Prentice-Hall.

Khan, N., Goldberg, D. E. and Pelikan, M. (2002) 'Multiobjective Bayesian optimization algorithm', *proceeding of the Genetic and Evolutionary Computation Conference (CECCO'2002).* San Mateo, CA:Morgan Kaufmann.pp 684

King, J. R. and Spackis, A. S. (1980) 'Scheduling: bibliography and review,' *International Journal of Physical Distribution and Materials Management*, 10, pp. 105-132.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983) 'Optimisation by Simulated Annealing', *Science*, 220, (4598), pp. 671-679.

Kita, H., Yabumoto, Y. and Nishikawa, Y. (1996) 'Multi-objective optimization by means of thermodynamical genetic algorithm', *In Proceeding of Parallel Problem Solving from Nature IV (PPSN - IV).* 504-512

Knowles, J. D. and Corne, D. W. (2000) 'Approximating the non-dominated front using the Pareto archived evolution strategy', *Evolutionary Computation Journal*, 8, (2), pp. 149-172.

Konak, A., Coit, D. W. and Smith, A. E. (2006) 'Multi-objective optimization using genetic algorithms: A tutorial', *Reliability Engineering & System Safety*, 91, (9), pp. 992-1007.

Koppen, M., Teunis, M. and Nicholay, B. (1997) 'NESSY - an evolutionary learning neural network', *In Proceeding of the Second International ICSC Symposium on Soft Computing SOCO'97.* 243-248

Koza, J. R. (1988) *Non-linear genetic algorithms for solving problems.* US papent 4935877.

Krishnakumar, K. (1989) 'Micro-genetic algorithms for stationary and non-stationary function optimization', *In SPIE Proceedings: Intelligent Control and Adaptive Systems*, pp. 231-240.

Krishnanand, K. and Ghose, D. (2005) 'Detection of multiple source locations using a glowworm metaphor with applications to collective robotics', *Proceedings of IEEE Swarm Intelligence Symposium.* 84-91

Kumar, A., Prakash, A., Shankar, R. and Tiwari, M. K. (2006) 'Psycho-Clonal algorithm based approach to solve continuous flow shop scheduling problem', *Expert Systems with Applications,* 31, (3), pp. 504-514.

Kumar, R. and Rockett, P. ((in press)) 'Improved sampling of the Pareto-front in multiobjective genetic optimizations by neo-stationary evolution: A Pareto converging genetic algorithm', *Evolutionary Computation*.

Kursawe, F. (1990) 'A variant of evolution strategies for vector optimization problems', *Parellel Problem Solving from Nature I (PPSN-1),* pp. 193-197.

Laumanns, M., Rudolph, G. and Schwefel, H. P. (1998) 'Spatial predator-prey approach to multi-objective optimization: A preliminary study', in. Vol. 1498, pp. 241.

Lederberg, J. (1959) 'Genes and antibodies: Do antigens bear instructions for antibody specificity or do they select cell lines that arise by mutation?', *Science,* 129, pp. 1649-1653.

Leung, K. S., Zhu, Z. Y. and Xu, Z. B. (1998) *Multiobjective optimization using non-dominated sorting in annealing genetic algorithms.* HongKong: Department of Geography and the Centre for Environment Studies, Chinese University of Hong Kong

Li, M. Q., Kou, J. S., Lin, D. and Li, S. Q. (2002) *The basic theory and application of genetic algorithms.* Science Press: BeiJing, CHINA.

Li, Y., Man, K. F., Tang, K. S., Kwong, S. and W.H., I. (2000) 'Genetic algorithm to production planning and scheduling problems for manufacturing systems', *Production Planning & Control,* 11, (5), pp. 443-458.

Loughlin, D. H. and Ranjithan, S. (1997) 'The neighbourhood constraint method: A multiobjective optimization technique', *In Proceeding of the Seventh International Conference on Genetic Algorithms.* 666-673

Loukil, T., Teghem, J. and Tuyttens, D. (2005) 'Solving multi-objective production scheduling problems using metaheuristics', *European Journal of Operational Research,* 161, (1), pp. 42-61.

Lu, H. and Yen, G. G. (2002) 'Dynamic population size in multiobjective evolutionary algorithms: ' *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on.* 1648-1653

Lu, H. and Yen, G. G. (2003) 'Rank-density-based multiobjective genetic algorithm and benchmark test function study', *IEEE Trans.Evol.Comput.,* 7, (4), pp. 324-343.

Luh, G.-C., Chueh, C.-H. and Liu, W.-W. (2003) 'MOIA: MULTI-OBJECTIVE IMMUNE ALGORITHM', *Engineering Optimization,* 35, (2), pp. 143 - 164.

Manne, A. S. (1960) 'on the job shop scheduling problem', *Operation Research,* 8, pp. 219-223.

Mansouri, S. A. (2004) 'A Multi-Objective Genetic Algorithm for mixed-model sequencing on JIT assembly lines', *European Journal of Operational Research,* 167, (3), pp. 696-716.

Mariano, C. E. and Morales, E. F. (2000) 'Distributed reinforcement learning for multiple objective optimization problems', California, CA, USA:IEEE, Piscataway, NJ, USA.pp 188-195

Mattfeld, D. C. and Bierwirth, C. (2004) 'An efficient genetic algorithm for job shop scheduling with tardiness objectives', *European Journal of Operational Research*, 155, (3), pp. 616-630.

Matyas, J. (1965) 'Random optimization', *Automation and Remote Control*, 26, (2), pp. 246-253.

Matzinger, P. (1994) 'Tolerance, danger and the extended family', *Annual Reviews in Immunology*, (12), pp. 991-1045.

Mercer, R. E. and Sampson, J. R. (1978) 'Adaptive search using a reproductive metaplan', *Kybernetes (The International Journal of Systems and Cybernetics)*, 7, pp. 215-228.

Michalewicz, Z. (1992) *Genetic algorithms + data structures = evolution programs.* Springer-Verlag: Berlin ; New York.

Miettinen, K. (1999) *Nonlinear multiobjective optimization.* Kluwer Academic Publishers: Boston.

Miller, B. L. and Shaw, M. J. (1996) 'Genetic algorithms with dynamic niche sharing for multimodal function optimization', *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on.* 786-791

Mitchell, M. (1998) *An introduction to genetic algorithms.* MIT: London, U.K.

Molyneaux, A. K., Leyland, G. B. and Favrat, D. (2001) 'A new clustering evolutionary multi-objective optimization technique', *Proceeding of the Third International Symposium on Adaptive Systems--Evolutionary Computation and Probabilistic Graphical Models.* Institute of Cybernetics, Mathematics and Physics, Havana, Cuba, 41-47

Moscato, P. (1989) *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts : Towards Memetic Algorithms.* Technical Report C3P 826 (Caltech Concurrent Computation Program)

Mucherino and Seref (2008) Available at: http://www.antoniomucherino.it/en/research.php (Accessed: 18 August 2010).

Mülhenbein, H. and Paaß, G. (1996) 'From recombination of genes to the estimation of distribution I. Binary parameters', *Lectures Notes in Computer Science: Parallel Problem Solving from Nature (PPSN IV)* 1411, pp. 178-187.

Muhlenbein, H., Gorges-Schleuter, M. and Kramer, O. (1988) 'Evolution Algorithms in Combinatorial Optimization', *Parallel Computing*, (7), pp. 65-85.

Murata, T. and Ishibuchi, H. (1994) 'Performance evaluation of genetic algorithms for flow shop scheduling problems', *Proceedings of the First IEEE International conference on Evolutionary Computation.* 812-817

Murata, T. and Ishibuchi, H. (1995) 'MOGA: multi-objective genetic algorithms', Perth, Aust:IEEE, Piscataway, NJ, USA.pp 289-294

Murata, T., Ishibuchi, H. and Tanaka, H. (1996) 'Multi-objective genetic algorithm and its applications to flowshop scheduling', *Computers & Industrial Engineering,* 30, (4), pp. 957-968.

Nagar, A., Haddock, J. and Heragu, S. (1995) 'Multiple and bicriteria scheduling: a literature survey', *European Journal of Operational Research,* 81, (88-104).

Nakrani, S. and Tovey, S. (2004) 'On honey bees and dynamic server allocation in Internet hosting centers', *Adaptive Behaviour,* 12.

Neef, M., Thierens, D. and Arciszewski, H. (1999) 'A case study of a multiobjective recombinative genetic algorithm with co-evolutionary sharing', 796-803

Nelder, J. A. and Mead, R. (1965) 'A simplex method for function minimization', *Computer Journal,* 7, pp. 308-313.

New, C. C. (1977) *Managing the manufacture of complex products: coordinating multicomponent assembly.* Business book: London.

Norwicki, E. (1999) 'The permutation flow shop with buffers: A Tabu Search approach', *European Journal of Operational Research,* 116, pp. 205-219.

Oliver, I. M., Smith, C. J. and Holland, J. R. C. (1987) 'A study of permutation crossover on the travelling salesmen problem,', *Proceeding of the Second International Conference on Genetic Algorithms and Their Applications.* 225-230

Ong, Z. X., Tay, J. C. and Kwoh, C. K. (2005) 'Applying the clonal selection principle to find flexible job-shop schedules', *Artificial Immune Systems, Proceedings,* 3627, pp. 442-455.

Onwubolu, G. C. and Mutingi, M. (1999) 'Genetic Algorithm for minimizing tardiness in flow-shop scheduling', *Production Planning and Control,* 10, pp. 462-471.

Osyczka, A. and Kundu, S. (1995) 'A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm', *Structural Optimization,* 10, (2), pp. 94-99.

Ousterhout, J. K. (1994) *Tcl and the Tk toolkit* Addison-Wesley: Reading, Mass. .

Pasupathy, T., Rajendran, C. and Suresh, R. K. (2006) 'A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs ', *The International Journal of Advanced Manufacturing Technology,* 27, (7-8), pp. 804-815.

Pedersen, M. E. H. (2010) *Tuning & Simplifying Heuristical Optimization.* PhD thesis. University of Southampton, School of Engineering Sciences, Computational Engineering and Design Group.

Pongcharoen, P. (2001) *Genetic Algorithms for Production Scheduling in Capital Goods Industries.* PhD thesis. University of Newcastle upon Tyne, U.K.

Pongcharoen, P., Hicks, C. and Braiden, P. M. (2004) 'The development of genetic algorithms for the finite capacity scheduling of complex products, with multiple

levels of product structure,' *European Journal of Operation Research,* 152, (215-225).

Pongcharoen, P., Hicks, C., Braiden, P. M. and Stewardson, D. J. (2002) 'Determining optimum Genetic Algorithm parameters for scheduling the manufacturing and assembly of complex products', *International Journal of Production Economics,* 78, (3), pp. 311-322.

Ponnambalam, S. G., V. Ramkumar and Jawahar, N. (2001) 'A multiobjective genetic algorithm for job shop scheduling', *Production Planning & Control,* 12, (8), pp. 764-774.

Ramasesh, R. (1990) 'Dynamic job shop scheduling: a survey of simulation research', *OMEGA,* 18, pp. 43-57.

Rastrigin, L. A. (1963) 'The convergence of the random search method in the extremal control of a many parameter system', *Automation and Remote Control* 24, (10), pp. 1337-1342.

Ray, T., Tai, K. and Seow, K. C. (2001) 'An evolutionary algorithm for multi-objective optimisation', *Engineering Optimization,* 33, (3), pp. 399-424.

Rechenberg, I. (1965) 'Cybernetic Solution Path of an Experimental Problem', *Royal Aircraft Establishment Library Translation*.

Reeves, C. R. (1995) 'A Genetic Algorithm for flow shop sequencing', *Computers & Operations Research,* 22, (1), pp. 5-13.

Robbins, H. and Monro, S. (1951) 'A Stochastic Approximation Method', *Annals of Mathematical Statistics,* 22, pp. 400-407.

Robert, G. S. (1983) 'Validating simulation models', *Proceedings of the 15th conference on Winter simulation - Volume 1.* Arlington, Virginia, United States:IEEE Press.pp

Rogers, A. and Prugel-Bennett, A. (1999) 'Genetic drift in genetic algorithm selection schemes', *IEEE Trans.Evol.Comput.,* 3, pp. 298-303.

Romero, C. E. M. and Manzanares, E. M. (1999) 'MOAQ and Ant-Q algorithm for multiple objevtive optimization problems', *Genetic and Evolutionary Computing Conference (GECCO 99).* San Francisco:Morgan Kaufmann.pp 894-901

ROSENBERG, N. (2003) 'Capital goods, technology and economic growth', *Oxford Journals, Oxford Economic Papers,* 15, (3), pp. 217-227.

Rothlauf, F. (2006) *Representations for genetic and evolutionary algorithms.* Springer: Heidelberg.

Rubinstein, R. Y. (1997) 'Optimization of Computer simulation Models with Rare Events', *European Journal of Operations Research,* 99, pp. 89-112.

Rudolph, G. (2001) 'Evolutionary search under partially ordered fitness sets', *In Proceeding of the International Symposium on Information Science Innovations in Engineering of Natural and Artificial Intellgent Systems (ISI 2001).* 818-822

Sareni, B. and Krahenbuhl, L. (1998) 'Fitness sharing and niching methods revisited', *Evolutionary Computation, IEEE Transactions on,* 2, (3), pp. 97-106.

Schaffer, J. D. (1985) 'Multiple objective optimisation with vector evaluated genetic algorithms', *In proceeding of the First International Conference on Genetic Algorithms and Their Applications.* Hillsdale, NJ:Lawrence Erlbaum Associates Inc.pp

Schlesinger, S. (1980) 'Terminology for model credibility', *Simulation*, (34), pp. 101-105.

Sebaaly, M. F. and Fujimoto, H. (1996a) 'Assembly sequence planning by GA search: A novel approach', *in Proc. Japan/USA Symp. Flexible Automat.: ASME,.* 1235-1240

Sebaaly, M. F. and Fujimoto, H. (1996b) 'A genetic planner for assembly automation', *in Proc. 1996 IEEE Int. Conf. Evol. Comput.. .* Piscataway, NJ:, 401–406.

Sebaaly, M. F. and Fujimoto, H. (1996c) 'Integrated planning and scheduling for job-shop assembly based on genetic algorithms', *in Proc. Int. Conf. Adv. Prod. Syst., APMS'96,.* Kyoto, Japan: Kyoto Univ., , 557–562.

Sebaaly, M. F. and Fujimoto, H. (1996d) 'A new approach for constrained GA-search: Assembly sequence planning—A case study ', *in Proc. Int. ICSC Symp. Intell. Indust. Automat. and Soft Comput.,* . Millet, Canada:Int. Comput. Sci. Conventions.pp 138–144.

Sefrioui, M. and Periaux, J. (2000) 'Nash Genetic Algorithms: Examples and applications', California, CA, USA:IEEE, Piscataway, NJ, USA.pp 509-516

Shah-Hosseini, H. (2009) 'The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm', *International Journal of Bio-Inspired Computation,* Vol. 1, Nos. 1/2, pp. 71-79.

Smith, S. F. (1980) *A Learning System Based on Genetic Adaptive Algorithms* PhD thesis. University of Pittsburgh.

Song, D. P., Hick, C. and Earl, C. F. (2002) 'Product due date assignment for complex assemblies', *International Journal of Production Economics,* 76, pp. 243-256.

Srinivas, N. and Deb, K. (1994) 'Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms', *Evolutionary Computation,* 2, (3), pp. 221-248.

Stewardson, D. J., Hicks, C., Pongcharoen, P., Coleman, S. Y. and Braiden, P. M. (2002) 'Overcoming complexity via statistical thinking: optimising Genetic Algorithms for use in complex scheduling problems via designed experiments', *Tackling Industrial Complexity: Ideas that Make a Difference.* Downing College, Cambridge, 9th-10th April. 275-290

Storn, R. and Price, K. (1997) 'Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces', *Journal of Global Optimization,* 11, (341-359).

Syswerda, G. (1991) *Scheduling Optimisation using Genetic Algorithms.* Van Nostrand Reinhold: New York.

Syswerda, G. and Palmucci, J. (1991) 'Application of genetic algorithms to resource scheduling', San Diego, CA, USA:Publ by Morgan-Kaufmann Publ, Inc., Palo Alto, CA, USA.pp 502

Tabucanon, M. T. (1989) 'Multiple criteria decision making in industry'.

Tan, K. C., Khor, E. F. and Lee, T. H. (1999) 'Evolutionary algorithms with goal and priority information for multi-objective optimization', *Proceeding of the 1999 Congress on Evolutionary Computation.* 106-113

Tan, K. C., Khor, E. F. and Lee, T. H. (2005) *Multiobjective evolutionary algorithms and applications.* Springer: London.

Tavakkoli-Moghaddam, R., Rahimi-Vahed, A. and Mirzaei, A. H. (2007) 'A hybrid multi-objective immune algorithm for a flow shop scheduling problem with bi-objectives: Weighted mean completion time and weighted mean tardiness', *Information Sciences,* 177, (22), pp. 5072-5090.

Tay, J. C. and Ho, N. B. (2008) 'Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems', *Computers & Industrial Engineering,* 54, (3), pp. 453-473.

Teerawut, T. (2008) *An Enhanced Grouping Genetic Algorithm for Optimising the Formation of Design Teams and Manufacturing Cells.* PhD Thesis thesis. Newcastle University, U.K.

Thapatsuwan, P., Chainate, W., Hicks, C. and Pongcharoen, P. (2010) 'Development of a stochastic optimisation tool for solving the multiple container packing problems', *The 16th international working seminar on production economics.* Innsbruck, Austria, 471-480

Todd, D. (1997) *Multiple criteria genetic algorithms in engineering design and operation.* Ph.D. thesis thesis. University of Newcastle upon Tyne, U.K.

Torn, A. and Zilinskas, A. (1989) *Global optimization.* Springer-Verlag: Berlin.

Tralle, D. (2000) *Analysing of Genetic Operations in Genetic Algorithms Applied to Optimisation of Manufacturing Systems.* University of Newcastle upon Tyne, U.K. (Undergraduate project).

Tseng, H. E., Chen, M. H., Chang, C. C. and Wang, W. P. (2008) 'Hybrid evolutionary multi-objective algorithms for integrating assembly sequence planning and assembly line balancing', *International Journal of Production Research,* 46, (21), pp. 5951-5977.

Tunnukij, T. and Hicks, C. (2009) 'An Enhanced Grouping Genetic Algorithm for solving the cell formation problem', *International Journal of Production Research,* 47, (7), pp. 1989-2007.

Veldhuizen, D. V. (1999) *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations.* Ph.D Thesis thesis.

Wang, X. and Cao, L. (2002) *Yi chuan suan fa : li lun, ying yong yu ruan jian shi xian* (Di 1 ban.) [2, 3, 344 p. : ill. ; 21 cm + 1 CD-ROM (4 3/4 in.)].

Watanabe, M., Ida, K. and Gen, M. (2005) 'A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem', *Computers and Industrial Engineering Selected Papers from the 30th International Conference on Computers and Industrial Engineering.* Elsevier Ltd, Oxford, OX5 1GB, United Kingdom,48 pp. 743-752.

Welp, E. G., Lindemann, U. and Endebrock, K. (1999) 'Design for recyclability of capital goods', *Environmentally Conscious Design and Inverse Manufacturing, 1999. Proceedings. EcoDesign '99: First International Symposium On.* 1-3 Feb.Digital Object Identifier 10.1109/ECODIM.1999.747715 pp 784-789

Whitley, D., Starkweather, T. and Fuquay, D. (1989) 'Scheduling Problems and Travelling Salesman: The Genetic Edge Recombination Operator', *Proceedings on the Third International Conference on Genetic Algorithms.* Los Altos, CA:Morgan Kaufmann Publishers.pp 133-140

Wikipedia (2010) *Metaheuristic.* Available at: http://en.wikipedia.org/wiki/Metaheuristic (Accessed: 18th Aug 2010).

Wilson, P. B. and Macleod, M. D. (1993) 'Low implementation cost IIR digital filter design using genetic algorithms', *In Proceedings of the IEE/IEEE workshop on Natural Algorithms in Signal Processing.*

Wirth, N. (1971) 'The Programming Language Pascal', *Acta Informatica,* Volume 1, pp. 35-63.

Wolpert, D. H. and Macready, W. G. (1995) *No free lunch theorems for search.* Technical Report SFI-TR-95-02-010 (Santa Fe Institute)

Wolpert, D. H. and Macready, W. G. (1997) 'No free lunch theorems for optimization', *IEEE Transactions on Evolutionary Computation* 1, (1), pp. 67-82.

Xie, W., Hicks, C. and Pongcharoen, P. (2010) 'An enhanced single-objective genetic algorithm scheduling tool for solving very large scheduling problems in capital goods industry', *The 16th international working seminar on production economics.* Innsbruck, Austria, March 1-5. 517-529

Xie, W., Hicks, C. and Pongcharoen, P. (2011) 'A multi-objective Genetic Algorithm scheduling tool for solving complex scheduling problems in the capital goods industry', *international Journal of Production Research (submitted on 16th February 2011).*

Xu, X. L., Wang, W. L. and Guan, Q. (2005) 'Adaptive immune algorithm for solving job-shop scheduling problem', *Advances in Natural Computation, Pt 2, Proceedings,* 3611, pp. 795-799.

Yang, X.-S. (2008) *Nature-Inspired Metaheuristic Algorithms.* Luniver Press.

Yang, X.-S. and Deb, S. (2009) 'Cuckoo search via Levy flights', *World Congress on Nature & Biologically Inspired Computing (NaBIC 2009).. IEEE Publication.* USA, 210-214

Yu, J., Yin, Y. and Chen, Z. (2006) 'Scheduling of an assembly line with a multi-objective genetic algorithm ', *The International Journal of Advanced Manufacturing Technology,* 25, (5-6), pp. 551-555.

Yu, J.-C. and Li, Y.-M. (2006) 'Structure representation for concurrent analysis of product assembly and disassembly', *Expert Systems with Applications,* 31, (4), pp. 705-714.

Zalzala, A. M. S. and Fleming, P. J. (1997) *Genetic algorithms in engineering systems.* Institution of Electrical Engineers: London, U.K.

Zandieh, M. and Gholami, M. (2009) 'An immune algorithm for scheduling a hybrid flow shop with sequence-dependent setup times and machines with random breakdowns', *International Journal of Production Research,* 47, (24), pp. 6999-7027.

Zhang, R. and Wu, C. (2008) 'An effective immune particle swarm optimization algorithm for scheduling job shops', *Iciea 2008: 3rd Ieee Conference on Industrial Electronics and Applications, Proceedings, Vols 1-3*, pp. 758-763.

Zhang, R. and Wu, C. (2010) 'A hybrid immune simulated annealing algorithm for the job shop scheduling problem', *Applied Soft Computing,* 10, (1), pp. 79-89.

Zitzler, E. and Thiele, L. (1998) *An evolutionary algorithm for multiobjective optimization: The strength Pareto approach.* Zurich, Switzerland: Computer Engineering and Network Laboratory (TIK), Swiss Federal Institute of Technology (ETH)

Zitzler, E., Deb, K. and Thiele, L. (2000) 'Comparison of multiobjective evolutionary algorithms: empirical results', *Evolutionary Computation,* 8, (2), pp. 173-195.

Zitzler, E., Laumanns, M. and Thiele, L. (2001) *SPEA-2: improving the strength Pareto evolutionary algorithm.* Zurich, Switzerland: