SINGLE MACHINE SCHEDULING WITH MODERN HEURISTIC TECHNIQUES

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of
Dokuz Eylül University
In Partial Fulfillment of the Requirements for
The Degree of Master of Science in Industrial Engineering Department

by
Güzin KAVRUKKOCA

July, 2003
İZMİR

Ms.Sc. THESIS EXAMINATION RESULT FORM

We certify that we have read the thesis, entitled **"SINGLE MACHINE SCHEDULING WITH MODERN HEURISTIC TECHNIQUES"** completed by GUZİN KAVRUKKOCA under supervision of ASST.PROF.LATİF SALUM and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst.Prof. Latif SALUM

Supervisor

Yard. Doç. Dr. Abdullah SEYRANKAYA

Committee Member

Yrd. Doç. Mehmet ÇAKMAKCI

Committee Member

Approved by the

Graduate School of Natural and Applied Sciences

Prof.Dr.Cahit Helvacı

Director

ACKNOWLEDGEMENTS

I am very happy to finish my Master Thesis which was prepared in a long time period. Although you see only my name on the cover, I am not alone and because of that I feel so lucky. I am great thankful to Assist. Prof. Latif SALUM, who is my supervisor from start to end by sharing his thoughts and experience. I am also grateful to Prof. Dr. Şevkinaz GÜMÜŞOĞLU, who contributed with her ideas. I would like to thank to my collegaue Res. Assist. Sabri ERDEM, who supported me with his deep knowledge and thoughts to construct such a successful study. Especially, thanks to Asst.Prof.Hüseyin AVUNDUK, and his collegues in PMS who provided me such a field of study and supported me with their practical experience by spending their time with me in shopfloor. I would like to thank to Assist. Prof. Yılmaz GÖKŞEN, Res. Assist. Aşkın ÖZDAĞOĞLU, Res. Assist. Aysun KAPUCUGİL for their help, support and tolerance. And finally thanks to my family who encouraged me with patience and tolerance during the time I wrote.

Güzin KAVRUKKOCA

ABSTRACT

Scheduling is an important problem in manufacturing environment. Many techniques have been generated for these kinds of problems, and in recent years, modern heuristic techniques are the mostly used of them. In this study, scheduling problems are examined in terms of modern heuristic techniques, and a genetic algorithm (GA) was used for developing a model. This model is a group-based scheduling problem on single machine - rotational molding process of plastic parts. Since the products have different shapes and volumes, being processed on the same machine, the major loss of the system is the set up times. Furthermore, the machine utilization is the other critical measure, because of differences in dimensions that have an impact on the number of parts produced in the same group. The scheduling algorithm developed especially focuses on these two critical objectives; therefore the problem can be defined as a multi-objective scheduling. The most important part of a genetic algorithm is the objective function, which is called "evaluation function" in GA terminology; the other properties of GAs are similar to each other. In literature, most evaluation functions are based upon only one objective and include relatively less information. In this study, discriminant analysis on normalized rank values generates the evaluation function. First of all, terms in objectives (variation of setup time within the group, total volume, process time/ setup time) are ranked according to the appropriate sequence, and to eliminate the effect of size of ranks, ranks are normalized by their own mean rank value. At the end of the analysis, sequence weights of each term of objectives are obtained, then these weights are used as coefficients of terms in the objective function. Finally, by using this function as priority, and the operators "crossover" and "mutation", the best sequences of groups are obtained and a successful schedule is developed. MS-Access and its Visual Basic Module are used for creation of the initial population and execution of the algorithm.

Keywords: Scheduling, heuristic techniques, genetic algorithm.

ÖZET

Çizelgeleme, üretim ortamlarında karşılaşılan önemli problemlerden biridir. Son yıllardaki eğilime bakıldığında, en çok uygulanan çizelgeleme modellerinin modern sezgisel yöntemler olduğu görülmektedir. Bu çalışmada, çizelgeleme problemleri modern sezgisel yöntemler açısından incelenmiş ve özellikle genetik algoritmalar (GA) üzerinde durulmuştur. Çalışmanın uygulama bölümünde, plastik parçaların döküm işlemini yapan tek makinalı bir üretim sisteminde grup bazlı çizelgeleme problemi için genetik algoritma ile desteklenen bir model sunulmaktadır. Farklı hacimlere ve biçimlere sahip olan, ancak üretim süreci gereği birarada işlenen bu ürünlerde en büyük kayıp hazırlık zamanlarından kaynaklanmaktadır. Boyutların farklılığı, birarada üretilebilen ürün sayısını da etkilemektedir ve bu durumda makina faydalanma oranları da etkili olmaktadır. Çalışmada anlatılan çizelgeleme modeli, bu iki kritik amaç üzerinde yoğunlaşmakta ve böylece çok amaçlı bir çizelgeleme modeli niteliği taşımaktadır. Genetik algoritmaların en önemli parçası olan amaç fonksiyonunda (GA terminolojisinde değerlendirme ya da uygunluk fonksiyonu) literatürde genelde tek amaç üzerinde durulmakta ve amaç değerlerine göre oransal olarak ifade edilmektedir. Bu da daha az bilgiyle çizelge geliştirilmesine neden olmaktadır. Bu çalışmada geliştirilen amaç fonksiyonu, ele alınan bileşenlerin normalize edilmiş sıra numaraları (toplam hazırlık süresi, grup hacmi, ürün sipariş miktarlarının grup içerisindeki değişkenliği) üzerinde ayırma analizi yapılarak elde edilmiştir. Analiz sonucunda her bir bileşenin grupların önceliği üzerindeki ağırlıkları elde edilmiş ve bu değerler katsayı olarak kullanılarak tek bir amaç fonksiyonu elde edilmiştir. Bu fonksiyonla birlikte çaprazlama ve mutasyon operatörleri kullanılarak, başarılı bir çizelgeye ulaşılmıştır. Başlangıç populasyonunun oluşturulması ve algoritmanın çalıştırılması için MS-Access ortamı ve Visual Basic modulünden yararlanılmıştır.

Anahtar Kelimeler: Çizelgeleme, sezgisel yöntemler, genetik algoritma.

CONTENTS

Chapter Three
MODERN HEURISTIC TECHNIQUES

Chapter Four
GENETIC ALGORITHMS

Chapter Five
PROPOSED APPROACH

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE
INTRODUCTION

1.1. Scheduling

Scheduling framework of this study contains general scheduling problems from the modern heuristic techniques viewpoint. Scheduling is a study that concerns the allocation of limited resources to the tasks, which can be in several forms, from service industry to manufacturing systems, or information processes (Pinedo, 1995).

Scheduling function should have a relationship with several other important functions in an enterprise. First of all it is affected by the production planning function, which handles medium and long-term planning for the entire enterprise. At this point, scheduling process should consider inventory levels, forecasts, and resource requirements to optimize, the product mix and resource allocation for long term period, then it focuses on unexpected events in the shop-floor such as machine breakdowns, or random processing times, and so on. All definitions of scheduling imply that it has a series of sequential steps or a routing. Schedules are also affected by product structure and job shop structure.

Objectives of general scheduling problems include optimization of due dates, flow times, and work center utilization. Most common of them cover minimization type objectives such as tardiness, earliness, setup times, flow times, etc.

Solution methods of scheduling problems are used under defined assumptions about process or product structure and also specific rules are generated for sequencing. Conventional scheduling problems are classified into different categories; static/dynamic, and deterministic/stochastic. Shop floor structure is also another classification way for scheduling such as single machine flow shop, open shop and job shop schedules, and these structures are the critical points for sequencing rules. Detailed information about the rules can be found in chapter 2.

Based on the constraints and information about scheduling problems considering shop floor and product structure, several algorithms have been developed. These algorithms may be mathematical models like linear programming or heuristic techniques that work according to the defined roots. If the problem is too complex and constraints can be constructed easily, then mathematical models can be used. But in general, it is difficult to model constraints and its objectives mathematically. Since scheduling is a type of short term planning, the scheduling should be obtained in a short time. Thus, mathematical models may be time consuming if the production environment has a complex structure. Therefore, modern heuristic algorithms have been generated. Even if most of these algorithms do not guarantee an optimal solution, they generate significant short-term plans near to the optimal one without spending to much waste time. A scheduling application of neural networks was implemented on parallel machines with additional heuristic rules (Park, 2000). A model called Potts was developed by a neural network algorithm that is useful for scheduling problems (Reeves, 1995). Hybrid models have been also used on scheduling models, one of the applications used such a model includes genetic algorithm and simulated annealing as a hybrid approach to schedule products with multi-level product structure (Kim, 1996). Most common hybrid models include genetic algorithm and neural network approaches.

1.2. Modern Heuristic Techniques

Modern heuristic techniques, also known as local search procedures usually, attempt to find a good schedule through a search. Basically, these approaches can be specified by three stages. The first stage generates an initial schedule, which is called "seed". The next stage exploits a mechanism of generating the method for selecting a schedule to replace the seed, which is usually generated by priority rules or other heuristic methods. The third stage is the selecting method, which decides the behavior of the search procedures.

There are many algorithms of heuristic techniques but most commonly used ones are investigated in this thesis. These are random search, hill climbing, tabu search, simulated annealing, neural network, and genetic algorithms.

Random search is an extremely basic method. It only explores the search space by randomly selecting solutions and evaluates their fitness. This is quite an unintelligent strategy, and is rarely used by it. Nevertheless, this method sometimes is worthwhile. It does not take much effort to implement it, and an important number of evaluations can be done fairly quickly. For new unresolved problems, it can be useful to compare the results of a more advanced algorithm to those obtained just with a random search for the same number of evaluations. *Hill climbing* is a standard local search procedure. It selects the first or the best improvement in the neighborhood to replace the seed. If the neighborhood of a solution is small, then it is reasonable and feasible to generate them all and select the best. If the neighborhood is very large, then to even generate them all might take a very long time. *Tabu search* is a neighborhood search based meta-heuristic technique that has the purpose of escaping local minima (Brandimarte and et al, 1995). This method tries to keep the search biased towards to good solutions. Tabu search has been used since late 1960s, and became an optimization technique for recent years. *Simulated annealing* is a local search method that uses thermal energy changing during the cooling system of a solid. Unlike tabu search, simulated annealing escapes from local minima in a probabilistic way. Besides accepting better schedules, it also, to a limited extent, accepts worse

schedules to replace the seed. Principle of *neural network* theory directly comes from biology. Neural networks are supposed to reproduce the same mechanisms as in the brain. They are able to solve very complicated problems in real time. They are widely used with excellent results for pattern recognition problems. Neural networks are also capable to learn the underlying mechanics of time series; they are often used for market dynamics in trading applications.

1.3. Genetic Algorithms

Genetic algorithms are search algorithms developed to explain and simulate the mechanisms of natural systems. In genetic algorithm applications, solution values are encoded into a structured string that represents a list of genes, which are referred to the characteristics of the solution of the defined problem. Genetic algorithms are one of the most common algorithms used for scheduling. Introduction of domain knowledge is most important source of high performance operators and led to clues for more effective representations and operators for scheduling problems.

Chapter 3 explains these search techniques in detail and one of them, called "genetic algorithms", are used for generating new sequencing rules in chapter 4 and 5.

1.4. Proposed Method

After theoretical framework for scheduling and modern heuristic techniques are introduced in the first four chapters, chapter 5 gives a proposed method, which is developed for a single machine scheduling problem using a genetic algorithm. Scheduling of the parts based on customer orders is one of the most important problems in the shop floor. All parts have very long process and setup times. Hence, short-term plans should be made effectively to meet the due dates. Main processes are carried out in rotational molding area, and this is the department that contributes to the manufacturing lead-time mainly. More than one part can be processed in a

cycle if volume and moment conditions are met. Setup times are too long and should be minimized. The scheduling problem is regarded as the grouping and sequencing the products to meet these objectives. The problem is about one of the rotational molding machines. There are six fixtures on this machine to place parts, thus at most 6 products can be processed in one cycle according to the volume and total available area constraints on the machine. Therefore, in each cycle, one load is selected to produce. The problem can be defined as "which group of products should be processed in which sequence on the machine?" The genetic algorithm developed for this problem produces successful results that has been obtained for a sequencing problem.

CHAPTER TWO
SCHEDULING

2.1. Introduction

For the general scheduling problem, tasks and resources are arranged or planned in order to reach one or more objectives. There are several scheduling definitions, but they all include similar constraints and objectives. The following are some of the definitions.

"Scheduling is a study that concerns the allocation of limited resources to the tasks, which can be in several forms, from service industry to manufacturing systems, or information processes" (Pinedo, 1995).
"Scheduling specifies the resources that each task needs at particular times. Any process that defines a subset of **What** × **When** × **Where** can be said to "do scheduling" (Parunak, 1990).

If these definitions are considered together, scheduling draws the following framework that includes Cartesian product of the questions:

1. There is a set of tasks (**What**) that must be done.

These tasks are not mutually exclusive. They include global tasks, "Make widget 3295" and primitive tasks from which the global tasks are constructed, "Order more screws".

2. There is a set of time intervals **(When)**, <start, stop>, where start < stop are real valued. Such an interval-based temporal logic has been extensively studied, and

permits reasoning about time points (start = stop) as a special case. One can associate with a task the time period during which it is performed.

3. There is a set of resources that tasks occupy as they execute **(Where)**. It includes not only geographically localized resources like machines, but also raw materials and other resources.

For any enterprise, a planner develops a scheduling function that covers all of the jobs in the firm or different functions are defined for the group of jobs. Scheduling function should have a relationship with several other important functions in the organization. First of all, it is affected by the production planning process, which handles medium and long-term planning for the entire organization. At this point, scheduling process should consider inventory levels, forecasts, and resource requirements to optimize at a higher level, the product mix and resource allocation for long term period, then it focuses on unexpected events in the shop-floor such as machine breakdowns, or random processing times, and so on. As known, planning is a whole system and the scheduling is one of the stages of this system. Figure 2.1 shows where scheduling is placed in the planning system as an information flow.

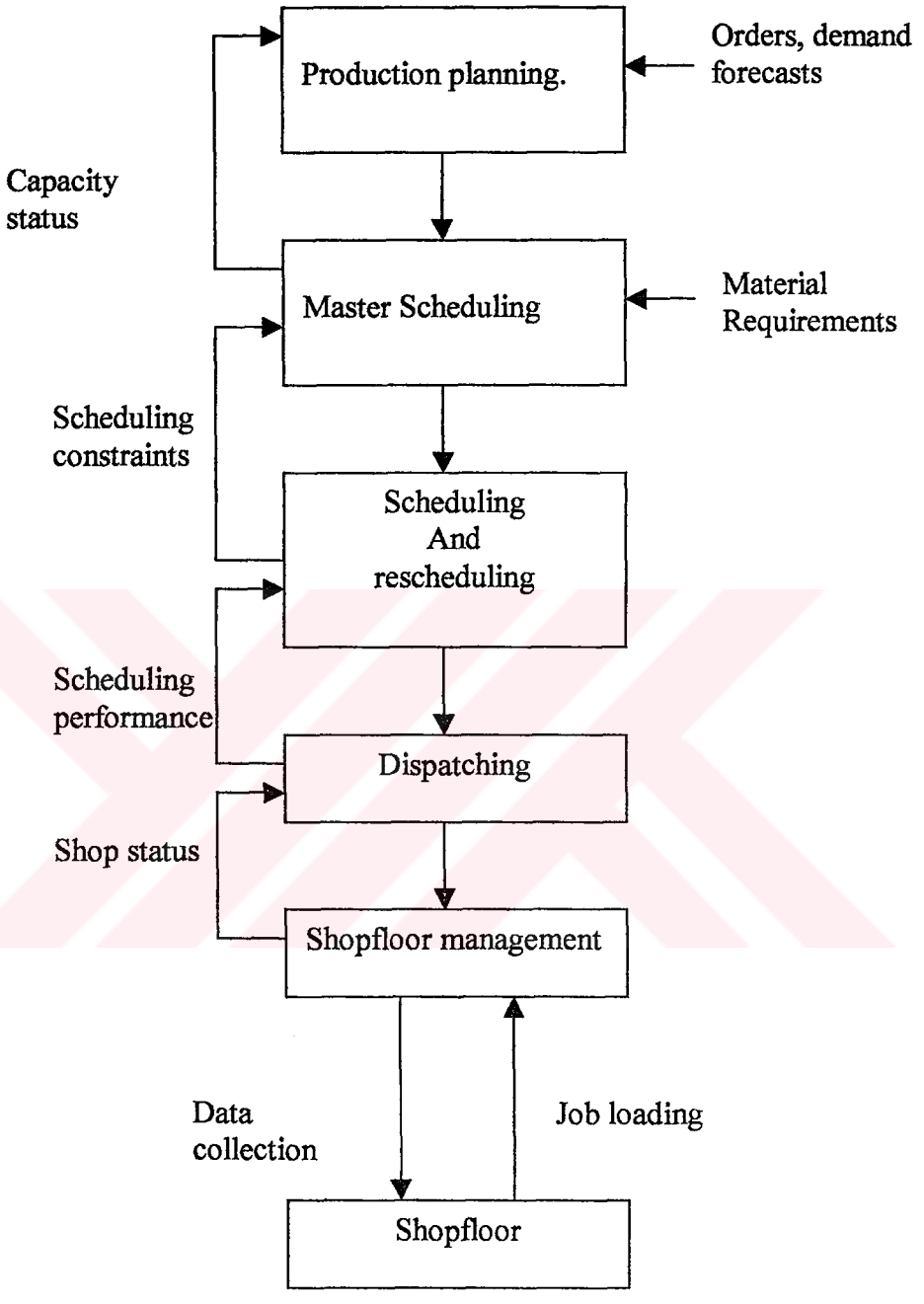


Figure 2.1. Information Flow Diagram in a Manufacturing System
Source: Scheduling Theory, Algorithms and Systems; Michael Pinedo, PrenticeHall, 1995.

2.2. Principles of Scheduling

All definitions of scheduling imply that it has a series of sequential steps or a routing. General plans or schedules have the following inputs related with these sequential steps:

- The sequence of operations,
- Necessary sequential constraints,
- The time estimates for each operation,
- Required resource capacities for each operation.

In addition to the definition above, scheduling is a group of processes that answers when an end item will be completed, and what jobs are to be completed during a specified time by the work center of interest.

2.2.1. Objectives of Scheduling Problems

There are many objectives applied to scheduling problems, but three primary goals are chosen in general:

Due dates: Late job completions are not wanted.

Flow times: This objective concerns to minimize the time a job spends in the system.

Work center utilization: This goal is concerned for fully utilization of capacity, because equipment and personnel are expensive.

These three major objectives are often in conflict. We can meet due dates if more capacity is provided.

The shop structure also lies in the scheduling framework. For example, flow shop is a repetitive manufacturing layout in which jobs go through a fixed sequence of the same routing steps whereas job shops have customized products, and each job has a unique routing steps. Another dimension of shop structure is about its *capacity*. Any

solution that works perfectly in 10-work center-shop, may not work well in 100-work center-shop of the same structure.

Product structure also affects the scheduling environment. Critical issues about product structures are:

- Existence of single part or assembly routings
- Type of processing time distribution.
- Alternative routings.
- Operation overlapping
- Lot sizes (variable/fixed)

One consideration is the extent to which capacities are fixed and or variable. This situation is similar to alternative routing issue. The extent to which capacity for a particular work center can be increased or decreased and the time delay to change the capacity both affect scheduling performance. Flexibility of the production environment is an important criterion for work center capacity. An additional issue in work center capacity is to focus attention on the bottlenecks. If the capacity of bottleneck work center can be utilized, then scheduling performance would be improved, too. At this point the cost of work-in-process parts and flow times should be kept at optimum level.

There exist several types of objective functions for scheduling problems developed according to the job characteristics and related assumptions. Most common job characteristics and objective functions were listed by (Pinedo, 1995) as in Table 2.1.

Table 2.1. Job Characteristics and Related Objective Functions

Job Characteristics	Objective Functions
• Preemption • Presence of additional resources • Non-zero release times • Jobs may wait/not wait • Due dates are soft/hard constraints	**Minimization of** • Flow time, • Total completion time, Total weighted completion time • Setup times • Lateness • Tardiness, Total tardiness, Total weighted tardiness • Earliness • Unit penalty • Weighted number of tardy jobs • Maximum lateness • Maximum makespan **Maximization of machine utilization.**

As seen in the table above, most common scheduling problems include minimization type objective functions, and consider penalties during production period, such as earliness, tardiness, setup times and so on.

2.2.2. General Assumptions

Because of the complexity of manufacturing and service environments, most of the scheduling problems are solved under some assumptions associated with job characteristics and objective functions. Classical assumptions of the scheduling problems are represented as follows:

1. Single parts and batches of parts are always treated as a single job.

This assumption may be appropriate for many situations, but in the case of large lots it may produce poor quality schedules.

2. Preemption is not allowed.

Preemption occurs when an operation is stopped and resumed at later time. This is not the case when scheduling tasks on computer systems.

3. Job cancellation is not allowed.

This means that all jobs are to be processed eventually, but may not be true. Therefore, defining priorities for jobs may be a good solution.

4. Processing times are independent of the schedule.

This assumption implies that setup times are sequence-independent and can be included in the processing time.

5. Work –in –process is allowed.

Jobs may wait in a queue until next machine required for processing is idle. In some metallurgical processes, this situation is not possible.

6. Machines are able to process one job at a time.

This is a correct assumption for many mechanical operations. Batch processors such as thermal treatment machine, PCB processors, etc violates the assumption.

7. Machines are always available.

There may be uncontrollable events, such as machine breakdowns; but it can mostly be prevented by a successful maintenance plan.

8. Machines are the only resources modeled.

In practice, it may be necessary to model additional resources such as transportation devices (AGVs, conveyors, cranes, etc.), fixtures, tools, or skilled workers.

9. Jobs may not start before its release date.

10. Setup times can be sequence dependent.

11. Jobs are known in advance.

This is the difference point between static and dynamic scheduling problems. In dynamic case new jobs may be arrived at unpredictable time.

12. Precedence constraints can be occurred.

13. Recirculation may occur/not occur.

14. Blocking of workstations is allowed/not allowed.

15. The problem is purely deterministic.

Scheduling problems are usually stochastic. For example, machine failures are significant degrees of uncertainty, but solutions of deterministic problems are keys of stochastic problems.

2.2.3. Sequencing and Related Rules

Sequencing is the major part of scheduling process in which the products are ordered according to the priorities (Chretienne, 1997). Generally, process times, setup times, and also due dates of the product changes, so this makes the sequencing more complex. Algorithms for sequencing differ by number of products, number of machines and objectives. Thus, for each sequencing algorithm, priority rules are developed by considering constraints and objectives. A large number of sequencing rules have appeared in research and in practice. Each could be used in scheduling jobs. Following rules are the most common among them:

- **R (random):** Pick any job in the queue with equal probability. This rule is often used as a benchmark to other rules.
- **FCFS (first come/first serve):** This rule is sometimes called to be "fair" in that jobs are in the order they arrive at the work center.
- **SPT (shortest processing time):** This rule reduces work-in-process inventory, average job completion time, and average job lateness.
- **EDD (earliest due date):** This rule seems to work well for criteria related with job lateness.
- **CR (critical ratio):** the ratio is used as a priority index (due date-now)/ (lead-time remaining).
- **LWR (least work remaining):** This rule is an extension of SPT rule in that it considers all processing time remaining until the job is completed.
- **FOR (fewest operations remaining):** It is another SPT variant rule that considers the number of successive operations.
- **ST (Slack time):** It is a variant of EDD rule that subtracts the sum of setup and processing times from the time remaining until the due date. The resulting value is called slack. Jobs are run in order of the smallest slack time.
- **ST/O (slack time per operation):** A variant of ST that divides the slack time by the number of remaining operations. Smallest value is the first in order.
- **NQ (next queue):** It is a different kind of rule; NQ is based on machine utilization. The idea is to consider queues at each of the succeeding work centers to

which the jobs will go and to select the job for processing that is going to the smallest queue.

- **LSU (least setup):** It is also another rule to pick the job that minimizes changeover time on the machine. In this way capacity of the work center is maximized.

Extensive research has addressed different sequencing rules performance. 39 different sequencing rules were tested using the same set of 10,000 jobs. Of the jobs from the first 400 and last 900 were not included in the results to eliminate startup and ending conditions. Table 2.2 and 2.3 show the results of this study for two criteria: average time in the system and variance of the time in the system (Vollman, 1997).

Table 2.2. Simulation Results For Various Sequencing Rules

Sequencing Rule	Average time in system	Variance of time
SPT	34.0	2,318
EDD	63.7	6,780
ST/O	66.1	5,460
FCFS	74.4	5,739
R	74.7	10,822

Source: (1997) Manufacturing Planning and Control Systems, Vollmann, McGrawhill, pp 125.

Table 2.3. Simulation Results For Other Criteria

Sequencing Rule	Average job lateness	Variance of job lateness	Percentage of jobs late
ST/O	-12.8	226	3.7
SPT	-44.9	2,878	5.0
EDD	-15.5	432	17.8
FCFS	-4.5	1,686	44.8

Source: (1997) Manufacturing Planning and Control Systems, Vollmann, McGrawhill, pp125.

In studies where the shop utilization varied, SPT was less sensitive to changes in work centers' capacity utilization than other sequencing rules.

2.3. Scheduling Systems

Conventional scheduling models and systems are classified into different categories hierarchically, and these categories are summarized in Figure 2.2.

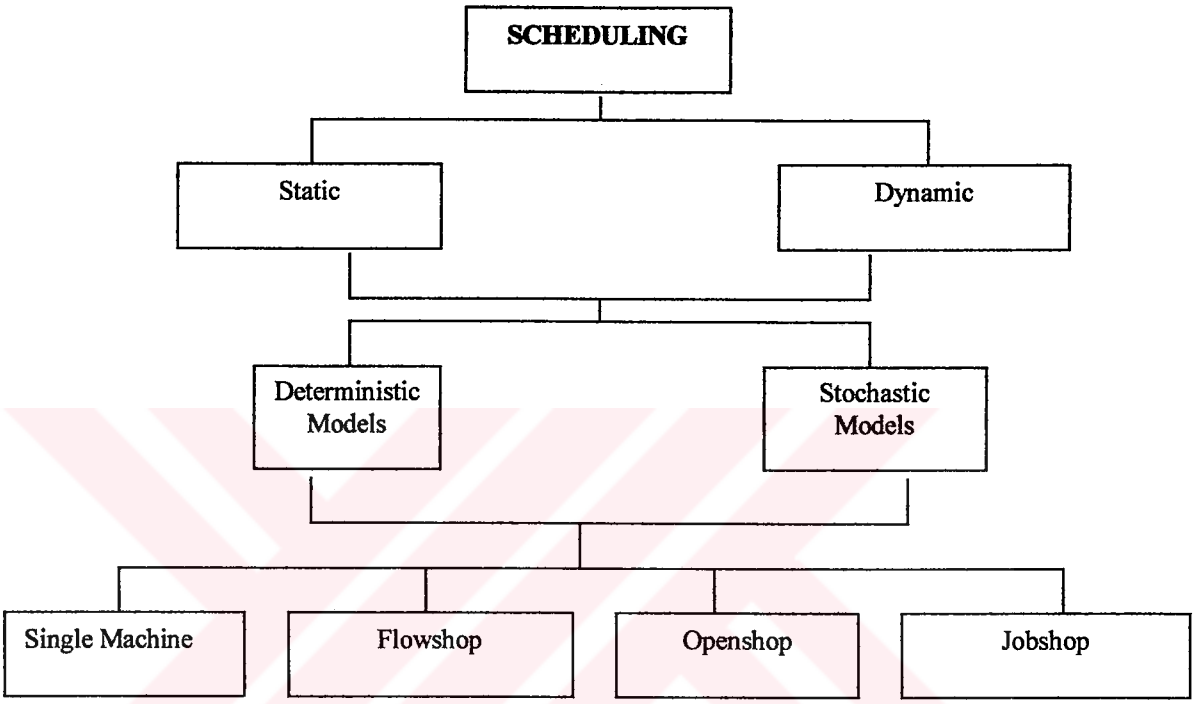


Figure 2.2. Classification of Scheduling

Next chapters will assume the hierarchy in Figure 2.2. Scheduling problems are first divided into two categories: static scheduling and dynamic scheduling.

2.3.1. Static Scheduling

The static scheduling problem includes fixed sets of jobs to be planned. Typical assumptions are the entire set of jobs arriving at the same time and all work centers being available at that time. Most static scheduling research uses a criterion called "minimum makespan", which means the minimum total time to process all jobs. This criterion indicates the flow time, but there are more criteria to be considered such as due dates, work center utilization, and so on.

Static scheduling research has been performed using deterministic processing times and stochastic processing times. Methods for dealing with deterministic times can be divided into those producing optimum results and those utilizing heuristic scheduling procedures. Since the computational difficulty increases exponentially with the problem size, optimization methods are only applicable to relatively small problems. Large scaled problems are usually treated with heuristic procedures called dispatching, or sequencing rules. These are logical rules for choosing which available job to select for processing at the particular work center.

2.3.2. Dynamic Scheduling

In dynamic scheduling problems, new jobs are added during production. Processing times may be deterministic or stochastic. The most common criteria for dynamic scheduling are average flow time, average work-in-process, or the number of jobs in the system and work center utilization. First applications were developed on single machine schedules concerning queuing theory, and later work extended the results to multiple machines. Simulation studies are most common for large-scale dynamic scheduling cases.

2.4. Deterministic and Stochastic Scheduling Models

Deterministic and stochastic scheduling problems have different principles and solution procedures according to the machine environment. Because of this reason, for each type of machine environment, deterministic and stochastic models are presented separately in the following sections.

Because scheduling procedures have great relationship with the layout of the manufacturing environment, basic layouts should be explained before the procedures. Figure 2.3 shows these layouts.

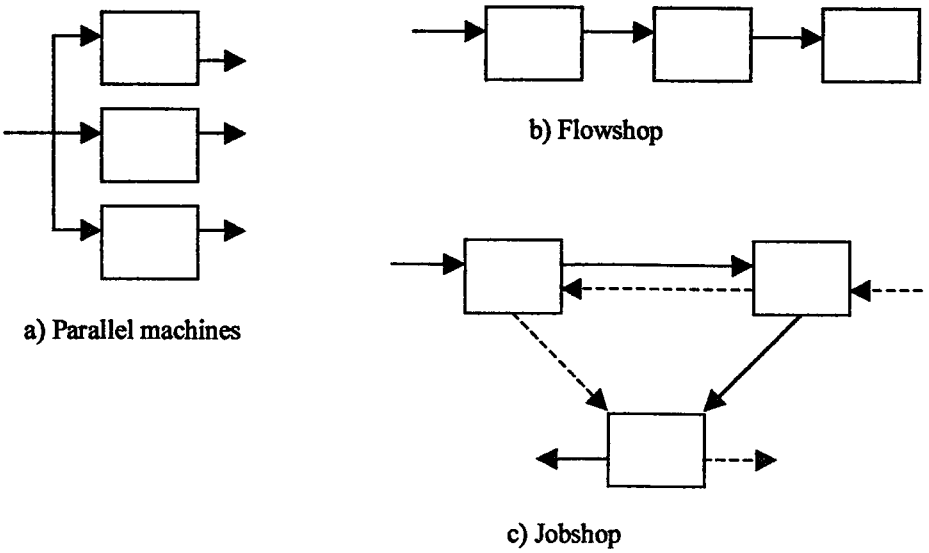


Figure 2.3. Basic Layouts of the Machine Environment.

2.4.1. Single Machine Models

Single machine scheduling can be considered that it has the least complexity process among the other type of scheduling, because single machine scheduling is a special case. In fact, single machine scheduling models include many hints for the other types; furthermore it provides new heuristics for the other environments with high complexity.

Research on single-machine scheduling has been largely based on the static problem of how the best schedule is obtained when a fixed set of jobs are available at the start of the scheduling period. If the objective is to minimize the makespan, it does not matter in which order jobs are run. In this case, the makespan equals the sum of all setup and run times for any sequence of jobs. However, if the objective is to minimize average time each job spends on the machine, then it can be obtained by sequencing jobs in ascending order according to their processing times. This sequencing method is also called "shortest processing time". This rule gives perfect schedules when the goal is to minimize average time in the system. It also performs well on the criterion of "average number of jobs in the system" and "average job

lateness". If the goal of the scheduling is to minimize the maximum lateness of a job, then the jobs should run in due date sequence.

Single machine models may be either deterministic or stochastic according to the properties of the manufacturing or service environment. Scheduling of bottleneck machines is considered as a single machine case, too. Real life problems are generally stochastic that they have machine breakdowns, unexpected orders, and random process times. But by the way they are solved; stochastic and deterministic models have similar solution methods (Chretienne, 1997). Especially in single machine cases, if the process times are randomly distributed, then the average of the distribution may be taken as a process time to solve the problem as a deterministic model. The important problem of single machine cases is the assumption whether preemption is allowed or not.

There are several methods that have been developed for scheduling problems such as linear programming, modern heuristic techniques, and so on. For example, branch and bound algorithms are used for many single machine-scheduling problems. These algorithms may be applied for whole solution of the problem or obtaining a partial solution. In the recent applications including branch and bound algorithms, specific heuristics were developed to optimize the lower and upper bound. For these algorithms, "lagrangian relaxation" was applied to obtain lower bound and upper bound, which consists of a two-phased heuristic procedure that combines a priority-dispatching rule with a local improvement procedure. Branch and bound algorithms were developed including these two bounds with dominance rules for independent jobs to minimize weighted earliness and tardiness with zero machine idle times (Liaw, 1999). Another single machine scheduling model was developed by using a hybrid algorithm consisting of simulated annealing and tabu search for just in time production system to obtain near optimal sequence in order to minimize the total cost for tardiness and earliness (Almeida, 1998). Composit heuristics are also used for single machine models, especicially for minimization of tardiness/earliness (Teressa, 1998).

2.4.2. Parallel Machine Models

Parallel machine models can be explained as special case of flexible flow shops that are used in practice so often. In this type of scheduling the most common objective functions are minimization of total completion time and minimization of makespan. Parallel machine models have two basic steps:

1. Which product will be produced at which machine?
2. Which sequence will they follow?

For most parallel machine environments, preemption of the machine is a subject of assumptions. Preemption means that any product that has not finished yet can be disposed at any time. Non-preemption is the inverse of this statement. Stochastic parallel machine problems have similar characteristics as given in the single machine case. Specific heuristics were developed for parallel machine scheduling including linear programming to determine starting time of the first job taking the earliness and completion time penalties as objectives for multiple machines (Biskup, 1999).

2.4.3. Flow Shops and Flexible Flow shops

In many manufacturing systems, most of the products follow the same route; therefore the machines are allocated in specified sequence according to this route. This kind of layout is called a flow shop. Assembly lines can be defined as flow shops. Printed circuit boards are common samples for flow shops.

For this type of allocation, buffer sizes between two machines are the most important factors. Scheduling should prevent blocking of machines. Buffer sizes can be assumed limited or unlimited. If a job has to be processed at each stage only on one of the machines, this kind of environment is called "flexible flow shops" or "compound flow shops". Minimization of the makespan is the common goal of the scheduler in this environment. Most of the flow shop problems include NP-hardness,

considering the same objective for flow shop scheduling problems, and NP-hardness was discussed for two-stage flow shop scheduling problem (Linn, 1999).

Developing scheduling procedures for two-machine problems is more complex than those for single-machine systems. In the two-machine case, there exist job routings, and a criterion should be selected and both machines should satisfy it. Some other assumptions are also needed for more realistic scheduling problems. For example, all jobs are available at the start of the schedule, and setup times are independent (Li,1999). To solve flow shop scheduling problems, a set of rules has been developed to minimize makespan in two -machine cases, but it is different from the single-machine case. If the total time to run the entire batch of jobs is to be minimized, this does not ensure that the average time each job spends in the system, or the average number of jobs in the system will also be minimized. The most common algorithm was developed by Johnson, which uses minimum makespan criterion, (Morton, 1993).

Several algorithms have been developed for flow shop scheduling problems. For example, a branch and cut algorithm was applied to flow shop-scheduling problems including n jobs and m machines to obtain the optimum sequence by considering minimization of maximum makespan. A Branch and cut algorithm is an enumeration technique to solve mixed integer programming, hence the optimization may need large number of iterations, or sets of expressions may contain lots of equations. Under these conditions, solution time increases exponentially (Mercado & Bard, 1998).

2.4.4. Job Shop and Open Shop Scheduling

Job shop scheduling is an important and complex activity in manufacturing. It can be described as a set of jobs composed of sequences of operations that are processed on a set of machines. It is a set of decision-making process, which allocates limited resources over time to perform a set of jobs to meet objectives. The complexity analysis for even a small job shop problems has lead to significant research in

heuristic rules. Many algorithms and rules have been applied for job shop scheduling problems since 1950s (Nahmias, 2001) to solve these kinds of problems, such as dispatching rules, search algorithms, and artificial intelligence (Vollman, 1997).

. Search algorithms are another approach that provides the strategies to explore the solution space efficiently, but sometimes their solution procedure can be slow and more complicated than the other approaches. Furthermore, search algorithms may only give the solution without telling how it is developed. Dispatching rule approaches use information related to a specific job or operation to arrange the next job to be processed on a machine. They provide a straightforward method to solve a problem using characteristics of jobs, operations, and machines. The procedure is easy to use and provides rapid solutions, but has limits for objective function or data elements. In the literature, dispatching rules are divided into two categories:

- Priority rules are based on information related to jobs, operations, or machines to determine the priority of resource allocation.
- Heuristic rules involve more complex considerations, such as machine load, alternative routing, and alternative operation. Shortest processing time is one of the examples for these rules.

Job shop scheduling generally includes dynamic processes so that the current schedule is updated according to the new orders and machine breakdowns. This concept is called rescheduling and simulation is a useful method to analyze all alternative schedules to meet the customer orders in order to change the schedule at any moment (Li & Shaw, 1998). Petri Net modeling is used for simulating the system to show all situations in the manufacturing environment as nodes and transitions. This type of modeling can be used for job shop scheduling problems to see the relationships among jobs and machines for both deterministic and stochastic processes, but another technique should be used to optimize the system such as linear programming (Song & Lee, 1998). Polynomial algorithms have been also developed for many scheduling problems. One of these studies consists of open shop scheduling considering minimization of total completion time and makespan. The purpose of the

study is to derive polynomial algorithms for related objectives (Kyparisis & Koulanas, 2000).

2.5. Semiactive, Active and Non-Delay Schedules

There exists an additional category for performance of a schedule. In a Gantt chart, there could be a gap, such that an operation can be moved earlier without delaying any other operation. If no machine is kept idle when there is an operation available for processing, this type of schedule is called no delay schedules. If no operation can be completed earlier by altering processing sequences on machines and not delaying any other operation, then it is called active schedule. Finally, if no operation can be completed earlier without altering the processing sequence on any of machines, the schedule is called semi active (Pinedo, 1995). The difference among these types of schedules can be seen on the Gantt charts (Figure 2.4 and 2.5).

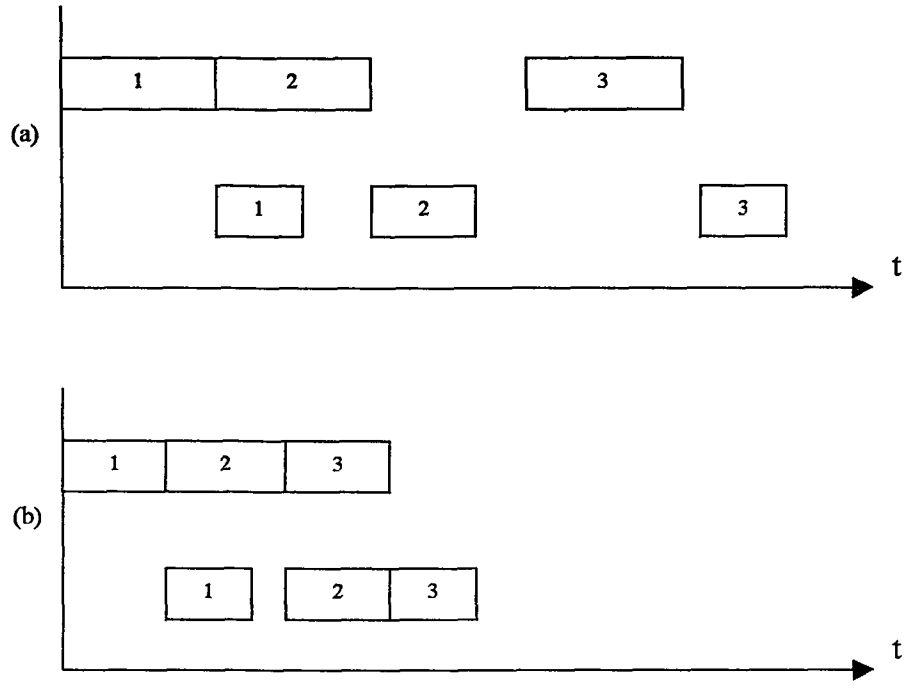


Figure 2.4. Nonsemiactive (a) And A Semi Active Schedule (b).
Source: (1995), Advance Models For Manufacturing Systems Management, Brandimarte, P; Villa, A.P:114

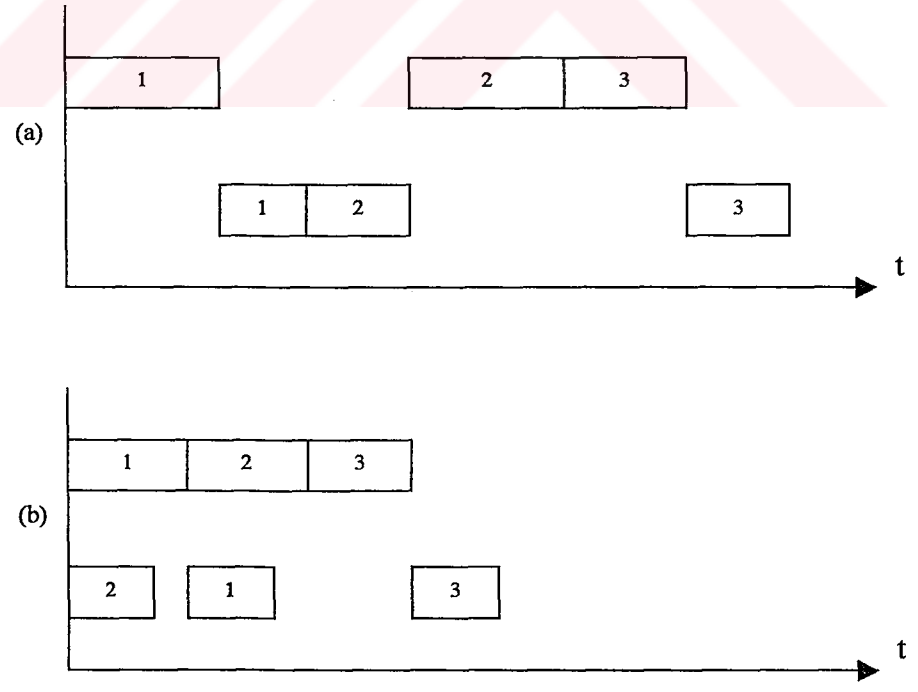


Figure 2.5. Non-semi Active but not Active (a) and the Corresponding Active Schedule (b).
Source: (1995), Advance Models for Manufacturing Systems, pp 210 Management, Brandimarte, P; Villa, A.p:114

CHAPTER THREE
MODERN HEURISTIC TECHNIQUES

3.1. Introduction

Modern heuristic techniques are also known as local search procedures and usually attempt to find a good schedule through a search. Basically, these approaches can be specified by three components:

- The first stage generates the initial schedule, which is called "seed",
- The next stage exploits a mechanism of generating the neighborhood. The method of selecting a schedule to replace the seed is usually generated by priority rules or other heuristic methods.
- The third stage is the selecting method, which decides the behavior of the search procedures.

In this chapter, these local search techniques, which are most common in literature, will be introduced firstly. Genetic algorithms, one of these search techniques, will be used for the implementation and explained in detail in chapter 3.

3.2. Random Search

Random search is an extremely basic method. It only explores the search space by randomly selecting solutions and evaluates their fitness. This is quite an unintelligent strategy, and is rarely used by itself. Nevertheless, this method is sometimes worthwhile to be tested. It does not take much effort to implement it, and an

important number of evaluations can be done fairly quickly. For new unresolved problems, it can be useful to compare the results of a more advanced algorithm to those obtained just with a random search for the same number of evaluations.

Random search has a few interesting qualities. However good the obtained solution may be, if it is not the optimal one, it can be always improved by continuing the run of the random search algorithm for long enough. A random search never reaches any stage such as a local optimum. Furthermore, theoretically, if the search space is finite, random search is guaranteed to reach the optimal solution. Unfortunately, this result is completely useless. For most of problems, exploring the whole search space takes far too long time.

3.3. Hill Climbing

Hill climbing is a standard local search procedure. It selects the first or the best improvement in the neighborhood to replace the seed. If the neighborhood of a solution is small, then it is reasonable and feasible to generate them all and select the best. If the neighborhood is very large, then to even generate them all might take a very long time. In this case it may be best to select the first improvement. One possible variation is selecting the best improvement from a fixed number of sampled schedules in the neighborhood. It is worthwhile noticing that there is no way to escape the first encountered local optimum since any move in the local neighborhood produces a worse schedule. Efficient methods exist for problems with well-behaved continuous fitness functions. These methods use a kind of gradient to guide the direction of search. *Stochastic Hill Climbing* is the simplest method of these kinds. Each iteration consists in choosing randomly a solution in the neighborhood of the current solution and retains this new solution only if it improves the fitness function. Stochastic Hill Climbing converges towards the optimal solution if the fitness function of the problem is continuous and has only one peak (*unimodal* function).

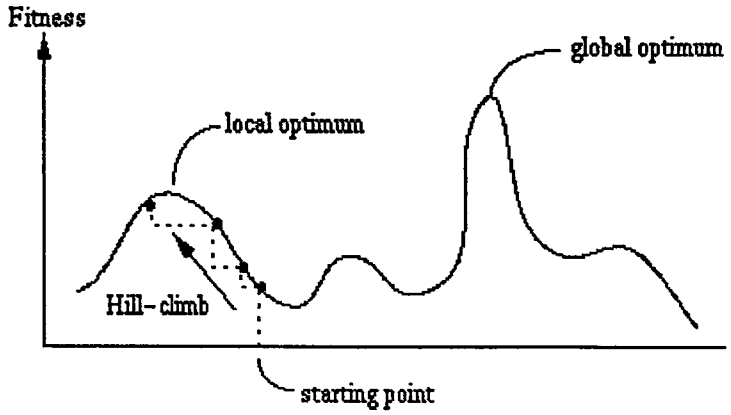


Figure 3.1. General Structure of Hill Climbing Method.
Source: www.epcc.ac.uk

On functions with many peaks (*multimodal* functions), the algorithm is likely to stop on the first peak it finds even if it is not the highest one. Once a peak is reached, hill climbing cannot progress anymore, and that is the problem when this point is a local optimum. Stochastic hill climbing usually starts from a random point. A simple idea to avoid from the first local optimal consists in repeating several hill climbs each time starting from different randomly chosen points. This method is sometimes known as iterated hill climbing. By discovering different local optimal points, it gives more chance to reach the global optimum. It works well if there are not too many local optimums in the search space. If the fitness function is very noisy with many small peaks, stochastic hill climbing is definitely not a good method to use. Nevertheless such methods have the great advantage to be really easy to implement and to give fairly good solutions very quickly .

3.4. Tabu Search

Tabu search is a neighborhood search based meta-heuristic technique that has purpose of escaping local minima (Brandimarte and et al, 1995). This method tries to keep the search biased towards good solutions. Tabu search has been used since late 1960s, and became an optimization technique for recent years. The basic ideas of tabu search have also been considered and singled out as the next decade of

Operations Research together with genetic algorithms and simulated annealing (Glover, 1993).

Tabu search allows the possibility that the selected solution is worse than the current schedule in a systematical way. It maintains a list of tabu restrictions to prevent the reversal, or sometimes repetition of certain moves. Every time one move is made, the reverse move is put at the top of the list and the restriction at the bottom is deleted. If a candidate move is tabu, the aspiration criteria are given an opportunity to override the tabu status. Thus the primary goal of the tabu restrictions is to prevent the search from becoming trapped at local optima. An important consideration in tabu search is the tabu list size. If the number of restrictions in the tabu list is too small, cycling may occur. If the number of restrictions is too large, they may be constrained improperly. The preferred tabu list sizes should lay in intervals related to problem dimensions.

The basic tabu navigation algorithm can be described as follows:

1. Choose the initial current solution and tabu list.
2. Evaluate the neighborhood of the current solution, and update the current solution with the best nontabu solution in the neighborhood.
3. Add some attribute of the new solution or of the applied perturbation to the tabu list.
4. If the maximum iteration number has been reached, stop the algorithm, otherwise update the parameters and go to step2.

For the tabu search method, important critics about the algorithm are,

- Relaxation criteria
- Selection of tabu attributes
- Length of the tabu list
- Restriction of the neighborhood
- Diversification

Tabu search technique can easily be used with specific heuristics or combined with another search technique. One of the applications including decomposition heuristic rule with tabu search analyzed resource flexibility by longest process time priority and results have shown that tabu search technique is one of the most effective local search methods in obtaining close approximations of optimal solution with modest computational effort (Daniels et al, 1999).

3.5. Simulated Annealing

Simulated annealing is a local search method that uses thermal energy changing during the cooling system of a solid. Unlike tabu search, simulated annealing escapes from local minima in a probabilistic way. Besides accepting better schedules, it also, to a limited extent, accepts worse schedules to replace the seed.

Simulated annealing was originally inspired by formation of crystal in solids during cooling. As discovered a long time ago by iron age blacksmiths, the slower the cooling, the more perfect is the crystal formed (Liu, 1999). By cooling, complex physical systems naturally converge towards a state of minimal energy. The system moves randomly, but the probability to stay in a particular configuration depends directly on the energy of the system and on its temperature. Gibbs law gives this probability formally:

$$p = e^{-\frac{E}{kT}}$$

Where E stands for the energy, k is the Boltzman constant and T is the temperature. In the mid 70's, Kirkpatrick, by analogy of these physical phenomena, laid out the first description of simulated annealing (Kirkpatrick, 1983). General steps in a basic simulated annealing method can be described as follows (Brandimarte, 1995).

1. Choose an initial solution, initial optimal value and initial temperature.
2. Choose a new optimal solution randomly from the neighborhood and compute the fitness value.
3. Calculate the acceptance probability (P).
4. Accept the new solution with probability of P. If this solution is accepted, replace with the old one.
5. If the termination condition is met, stop the algorithm, otherwise update the parameters and go to step2.

The phases of simulated annealing can be shown in Figure 3.2.

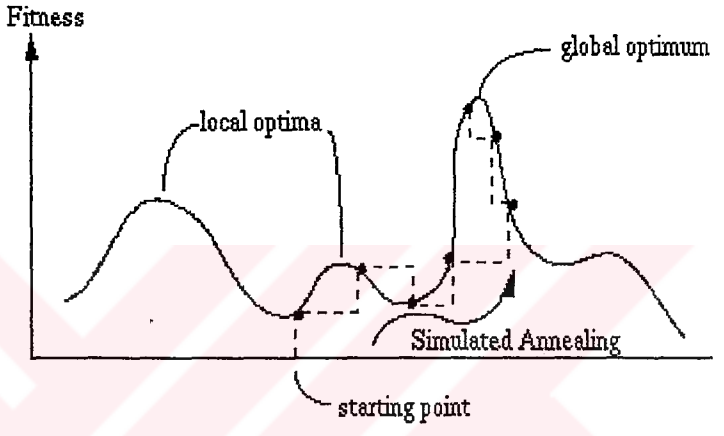


Figure 3.2. Phases of Simulated Annealing.
Source: http://epcc.ed.ac.uk

As in the stochastic hill climbing, the iteration of the simulated annealing consists of randomly choosing a new solution in the neighborhood of the actual solution. If the fitness function of the new solution is better than the fitness function of the current one, the new solution is accepted as the new current solution. If the fitness function is not improved, the new solution is retained with a probability:

$$p = e^{\frac{-(f(y)-f(x))}{kT}}$$

where $f(y)$-$f(x)$ is the difference value of the fitness function between the new and the old solution.

The simulated annealing behaves like a hill climbing method but with the possibility of going downhill to avoid being trapped at local optima. When the

temperature is high, the probability to deteriorate the solution is quite important, and then a lot of large moves are possible to explore the search space. The more the temperature decreases, the more difficult it is to go downhill; the algorithm tries to climb up from the current solution to reach a maximum. When temperature is lower, there is an exploitation of the current solution. If the temperature is too low, no deterioration is accepted, and the algorithm behaves just like a stochastic hill climbing method (Lim, 1997). Usually, the simulated annealing starts from a high temperature, which decreases exponentially. The slower the cooling, the better it is to find good solutions. It even has been demonstrated that with an infinitely slow cooling, the algorithm is almost certain to find the global optimum. The only point is that infinitely slow cooling often requires near infinite time. The main difficulty of simulated annealing consists in finding the appropriate temperature decrease rate to obtain a good behavior of the algorithm. Simulated Annealing by mixing exploration features such as the random search and exploitation features like hill climbing usually gives quite good results.

One advantage of these three local search procedures is that they do not have to know much about the structural properties of the problem, such as derivatives of the objective function. This feature is attractive for problems in which derivatives of the objective function are unknown. If the derivative of the objective function can be determined, the advantage vanishes and these three procedures typically become inferior to gradient-type methods, since they require much computing time to evaluate each new solution during a local search (Wall, 1996).

3.6. Constraint-Based Methods

A constraint satisfaction problem is defined by a set of variables and a set of constraints that can simultaneously be assigned to these variables. Constraint satisfaction problems with an objective function should be optimized subject to the problem constraints. The resource constraints are that each machine performs at most one task at any given time. Other constraints, which may affect scheduling, are

release times, due dates, transfer times, setup times, and resource availability (Nadeen, 2000).

Some constraints are relaxable, such as due dates, frequency of tool changes, inventory levels, etc. To solve the constraint satisfaction problems, a schedule is incrementally built by one scheduling decision. An important concept in constraint-based search methods is constraint propagation. Whenever a variable is instantiated, a new search state is created, where new constraints are used to deduce that some other unexplored values become inconsistent with the decision and must be pruned.

Therefore constraint propagation reduces the amount of search needed to generate a schedule. However, constraint propagation cannot detect all inconsistencies during schedule construction. If a partial schedule is reached that cannot be completed without violating a problem constraint, one or several earlier assignments need to be altered. Such a process is called backtracking. Because the problem is NP-complete, backtrack search may require exponential time in the worst case. Research in constraint satisfaction problems has produced several techniques to improve the efficiency of the backtrack search.

3.7. Genetic Algorithms

Genetic algorithms (GA) are search algorithms developed to explain and simulate the mechanisms of natural systems. In genetic algorithm applications, solution values are encoded into a structured string that presents a list of genes, which are referred to as the characteristics of the solution of the defined problem. A GA method is one of the major chapters of this study, and detailed presentation is given in chapter 4.

3.8. Neural Networks

Principles of neural network theory directly come from biology. Neural networks are supposed to reproduce the same mechanisms as in the brain. They are able to solve very complicated problems in real time. They are widely used with excellent results for pattern recognition problems.

Neural networks are also capable to learn the underlying mechanisms of time series; they are often used for market dynamics in trading applications. Precisely, Neural networks are not an optimization tool like GA, but a kind of interpolation tool. A neural network is made of nodes, called, by analogy with the brain, neurons, connected by weighted edges, called synapses. Typically, the nodes are divided into different layers; there is an input layer and an output layer, and hidden layers in between. That forms the topology of the neural network. During a training phase, weight over the synapses is setup in order to pass knowledge through the network (Lee, 2000).

There are two different kinds of architectures in neural network modeling, "feed-forward" and "feed-back". In feed-forward networks signals are processed from a set of input units in the bottom to output units in the top, layer by layer using the local updating rule of specified equation. In the feed –back networks, the synapses are bi-directional. Activation continues until a fixed point has been reached (Reeves,1995).

The point is that good neural networks are difficult to design. Many of the basic principles governing information processing are hard to understand, and there are a lot of complex interactions among network units. Hence, a lot of work has been done to apply GA in neural network design. GA can be used in different ways: for topology optimization, for training process, or for optimizing some control parameters.

Some work has also been done for training neural networks with genetic algorithms. The problem is formulated as a weighted optimization problem, usually using the inverse mean squared error as a fitness measure. But so far, methods of these kinds have been outperformed by specific methods originally used to train neural networks like the classical back propagation.

CHAPTER FOUR
GENETIC ALGORITHMS

4.1. General Definitions

As mentioned in the previous chapter, genetic algorithms are a local search method using parameters and operators with a specified coding system based on evolutionary processes. The objective of the solution is its performance in a survival competition. Genetic algorithms start with a population of candidate solutions. These solutions compete against each other, and survivors of this competition reproduce to form new solutions by exchanging partial characteristics of their solutions. After generations, these survivors become similar to each other. Most common operators of genetic algorithms are mutation and crossover. These operators have different implementation types that are determined by heuristics of the decision maker. Among the other heuristic techniques, genetic algorithms differ with the following properties (Goldberg, 1989):

- Genetic algorithms work with a coding of the parameter set, not the parameters themselves.
- Genetic algorithms search from a population of points not a single point.
- Genetic algorithms use payoff (objective function) information, not derivatives or other auxiliary knowledge.
- Genetic algorithms use probabilistic or deterministic transition rules.

The first development of genetic algorithms had been laid down based on the *schema theorem.* Genetic algorithms work by building blocks in the chromosome,

blocks that should be present in the optimal solution. With selection and recombination, such blocks naturally emerge and are passed to the next generation. It is called the building block hypothesis. Many to explain the power of genetic algorithms have granted this demonstration. But it may lack of consistency, mainly because too many suppositions are done (Whitley, 1993). Some limitations of schema theorem has even been claimed that the schema theorem has no implications for how well a genetic algorithm is performing. Performance of genetic algorithms is due to the correlation between parent and offspring fitness as he states in his version of Price's Theorem (Altenberg, 1995). Unfortunately, the recent No Free Lunch theorem has shaken most of those attempts to explain the power of genetic algorithms. The point is that Genetic algorithms cannot work properly if it does not include any domain knowledge (Wolpert & Marcready, 1995).

Most early applications were in the realism of artificial intelligence-game playing and pattern recognition. Some research was carried out on function optimization, but it is only recently that application to problems of interest to the Operational Research community such as the types of combinatorial optimization problems has become more widely reported. Basic concepts of genetic algorithms are (Goldberg, 1989):

- Genetic algorithms can be understood as the intelligent exploitation of a random search.
- GA approach is far more flexible and provides a general framework for a wide variety of problems.
- GAs give an interesting survey of the practical work carried out in this area.

The name GA originates from the analog between the representation of a complex structure by means of a vector of components and the idea familiar to biologists, of genetic structure of a chromosome.

Explanations given below indicate new concepts for problems except biology. Before the details of a general genetic algorithm, it would be useful to present these concepts briefly:

Crossover: An exchange of selections of parent's chromosome.

Mutation: A random modification of the chromosome.

Chromosomes = Solutions = Vectors.

Genes = Variables.

Alleles = the possible values of a variable.

Locus = Position of a variable in a string.

Genotype = the actual structure of a chromosome. (Coded string, which is processed in the algorithm).

Phenotype = The physical expression of the structure of the chromosome (Parameters).

Using the terminology given above, a Genetic algorithm works according to the following steps (compiled from www.epcc.ac.uk):

1. Start with an initial time.
2. Initialize a usually random population of individuals.
3. Initial population.
4. Evaluate fitness of all initial individuals of the population.
5. Evaluate the initial population.
6. Test for termination criterion (time, fitness, etc.).
7. Select a sub-population for offspring production.
8. Recombine the "genes" of the selected parents.
9. Recombine.
10 Perturb the mated population stochastically.
11. Mutate population.
12. Evaluate its new fitness.
13. Evaluate new generation.
14. Select the survivors from the actual fitness.
15. Do until termination.

Before starting the algorithm, some concepts should clearly be defined; the concepts given below construct the heuristic substructure of the genetic algorithm:

- Representation of chromosomes.
- Fitness Function.
- Definitions of operators (mutation, cross-over, etc.)
- Selection of parents.
- Termination criteria.

These concepts are introduced and explained in the following parts of this chapter.

4.1.1. Representation

Representation of chromosomes (members of the population) is related with the type of required solution. Representation may completely show the final solution, or present some part of the solution as any step of the final solution due to the definitions of the chromosome, genotype and phenotype of genes. Genotype is the information of the gene on the chromosome and phenotype is the physical meaning of the genotype. Strings generally represent chromosomes with several locuses including gene information. Any chromosome is divided into locuses, and each locus may include one or more gene information. The format of the representation changes according to the type of the algorithm and the problem (Mitchell, 1996).

Most common representation types are presented as follows:

- **Binary strings:**

Classical GAs use a binary string to represent a potential solution to a problem. Such a representation is not naturally suited for ordering problems such as the tabu search problems, because no direct and efficient way has been found to map possible solutions 1:1 onto binary strings.

Binary strings contain 1 or 0 value for each gene. Each locus is defined for the solution; it may be machine definition or product definition, or corresponding binary value of any quantitative result. For example, if a chromosome is defined to determine a machine group, then each locus may correspond to a machine in the

environment, and if the gene value is 0, it means that the machine does not belong to the group. If the value is 1, then the machine is placed in the group.

- **Non-binary strings**

Nonbinary string representations may contain any value of a gene. The gene value may be a letter, number or specific symbols, or their mixtures, and their explanations correspond to the phenotype of each gene. Non-binary representations are frequently used for the last decay, because many attributes can be assigned to the string by using nonbinary codes such as product codes and sequence numbers (Reeves, 1997).

- **Hierarchical representation**

Branches of a tree represent chromosomes. Especially for genetic programming approaches, hierarchical representation is most useful. Genetic programming is a software development process using genetic algorithms. Tree representation may be binary or nonbinary based on a problem. Automatically defined functions are also used for hierarchical representation to solve mathematical functions and to analyze performance of computer programs (O'Reilly, 1995).

4.1.2. Initial Population

Before starting the evolution process, an initial generation should be developed. It may be random, or generated according to a specific algorithm, such as knowledge-based scheme. Another type of initial population generator was developed by building block hypothesis (Goldberg, 1989). In addition, neural networks was applied to generate initial population to be used in genetic algorithms (Reeves, 1995).

4.1.3. Fitness Function

GA works by maintenance of a population of M chromosomes (potential parents) whose fitness values have been calculated. Each chromosome encodes a solution to the problem. Fitness value is related to the value of the objective function for the

solution. The fitness function is fully defined by the decision maker. If scheduling problems are considered, earliness/tardiness of jobs, or average earliness/tardiness of jobs, setup times, total makespan can be defined as a fitness function (Reeves, 1997).

4.1.4. Selection Process

One of the concepts of genetic algorithms is the selection process for evolution. After one decides the representation system, the next step is to determine the selection process. The selection scheme involves survival competition and reproduction mating. The selection scheme controls the growth speed of good solutions. There are several selection schemes available in genetic algorithm research.

Proportion selection chooses individuals for birth according to their objective function value. The probability of selection of an individual from a population is the ratio of its objective or evaluation function value over the expected objective value of the whole generation.

Fitness Proportionate With Roulette Wheel And Universal Sampling Method

This method is used by dividing each fitness value of individuals by average fitness value of current generation. This method was first used with sampling using roulette wheel through the following steps:

1. Sum the total expected value of individuals in the population and call this sum *T*.
2. Repeat *N* times:

- Choose a random integer "*r*" between 0 and *T*.
- Loop through the individuals in the population, summing values until the sum is greater than or equal to "*r*".

The individual whose expected value puts the sum over this limit, is the one selected. Third method has major problems when population size is small, since the

actual number of offspring allocated to an individual is often far from its expected value (Bierwirth & Mattfeld, 1999).

Sigma Scaling or sigma truncation has been developed by the following formula (Goldberg, 1989). Expected fitness value of population is the function of the population mean and standard deviation.

$$ExpVal(i,t) = \begin{cases} 1 + \dfrac{f(i) - \bar{f}(t)}{\sum \sigma(t)} & if\ \sigma(t) \neq 0 \\ 1 & if\ \sigma(t) = 0 \end{cases}$$

When population is typically more converges and standard deviation is lower, then the fitter individuals will stand out more, allowing evolution to continue.

Elitism is an addition to many selection methods that forces the genetic algorithm to retain some number of the best individuals of the previous population. Elitism was applied to save best individuals so that they are not destroyed or lost in the crossover or mutation processes (Mitchell, 1996).

Boltzman Selection. Sigma scaling keeps the selection pressure more constant over a run, but often different amounts of selection pressure are needed at different times in a run. This allows selecting worse individuals to reproduce better offsprings. Bolzman is similar to simulated annealing approach in which varying temperature controls the rate of selection.

Rank Selection. Rank selection is an alternative procedure whose purpose is to prevent the genetic algorithm from quick convergence. To implement this method individuals in the population are ranked according to the fitness values, and the expected value of each individual depends on its rank rather than on its absolute fitness value.

Rank selection, which is obtained by sorting the whole population from best to worst. Different from proportion selection, ranking selection defines a non-increasing assignment function to assign the number of copies that each individual contributes to a breeding pool, and then proportion selection is performed.

For instance, linear ranking method consists of the steps below:

1. Each individual in the population is ranked in the increasing order of the fitness function value, from *1* to *N*. The user chooses the individual with maximum expected value with rank *N*, with $maximum \geq 0$.
2. The expected value is calculated by the following formula:

$$ExpVal(i,t) = Min + (Max - Min)\frac{rank(i,t) - 1}{N - 1}$$

Fitness is a function that shows the individual performance as a solution. In other words, this function indicates the objective function value of the related individual, but if there exist more than one fitness value depending on the number of objectives, then this rank procedure should be modified.

An alternative solution for this case is developed in chapter 5, by using normalized rank values to develop the fitness function. Computing ranks for each objective at each phase of grouping and sequencing is relatively time consuming. Therefore, discriminant coefficients are used for each objective function value as a specific fitness function as explained in chapter 5.

Tournament Selection. Fitness –proportionate selections require two phases: The first phase computes the mean of the fitness function and the second one computes the standard deviation of the fitness function. Rank scaling requires sorting, which may be time consuming. Tournament selection is similar to rank selection in terms of the selection pressure. Two individuals are chosen at random from the population. A random number *"r"* is then chosen between (0,1). The fitter of two individuals is

selected to be a parent. The two are returned to the original population and can be selected again (Goldberg, 1989).

Steady –State Selection. In seady-state selection method, only a few individuals are replaced at each generation, whereas all population is changed in the other selection procedure except the elitist method. This type of selection is often used for rule-based systems (Mitchell, 1996).

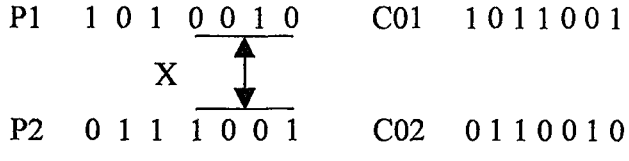
4.1.5. Genetic Operators

Genetic algorithms generate new solutions through duplication, recombination and mutation. Duplication generates a copy of genes to prepare for a later recombination process. Recombination, called crossover, and mutation are the key mechanisms by which genetic algorithms explore the solution space.

Crossover

Crossover generates new offspring by exchanging partial information from the parents. Like other search algorithms, a crossover operator provides a new search direction. Methodologies of genetic operators are relative to chromosome representation and the characteristics of the genes, because these methodologies must keep solutions feasible.

Choosing a crossover point X at random or due to a specified point selection matches parents; the offspring consisting of the pre-X section from one parent follows the post-X section of the other. For example;



Where, *P1* and *P2* indicate parent-one and parent-two, and also *C01* and *C02* indicate child-one and child-two, respectively. Pre-X section means the first *n* genes

before crossover point (or X), whereas Post-X section indicates the genes after crossover point (or X).

One of the existing populations is chosen at random, and replaced by one of the offsprings. This reproductive plan is repeated, as much time as desired. In another version of this procedure, parents are chosen in strict proportion to their fitness values (not probabilistically) and whose population is chosen a block after every set of *M* trials, rather incrementally. There are various parametric choices, and it is also quite possible that the way initial population is chosen will have a significant impact on the results (Reeves, 1995).

For sequencing problems, several operators are available such as ***partially mapped crossover***, which copies a section of genes from one parent and the rest by position-wise exchanging (Miller, and Goldberg, 1995). Another type of crossover process is ***Order-based crossover,*** which copies the chromosome from a parent, randomly selects a set of genes, and reorders these genes with the same relative sequence in the other parent. Effective crossover is a method for replicating patterns of genes with good objective functions (Reeves, 1995).

Mutation

Mutation operators change the value of a gene to keep the solution diversity. It can include swapping procedure, which clears some genes from the chromosome, or exchange the genes with new ones that do not provide diversity of the offsprings. Mutation can be defined as a specific function value, or can be run as an additional crossover operator.

Genetic algorithms stops at a pre-determined **termination point** or a rule. This rule can be defined by the number of iterations, a time interval or a criterion.

4.2. Parallel Genetic Algorithms

One of the major aspects of GA is their ability to be paralleled. Indeed, because natural evolution deals with an entire population and not only with particular individuals, it is a remarkably highly parallel process. Except in the selection phase, during which there is competition between the individuals, the only interactions between members of the population occur during the reproduction phase, and usually, no more than two individuals are necessary to engender a new child. Otherwise, any other operations of the evolution, in particular the evaluation of each member of the population, can be done separately. Thus, nearly all the operations in a genetic algorithm are implicitly parallel.

Parallel genetic algorithms (PGA) simply consists in distributing the task of a basic genetic algorithm on different processors. As those tasks are implicitly parallel, little time will be spent on communication and thus, the algorithm is expected to run much faster or to find more accurate results. It has been established that GA efficiency to find optimal solution is largely determined by the population size. With a larger population size, the genetic diversity increases, and so the algorithm is more likely to find a global optimum. A large population requires more memory to be stored; it has also been proved that it takes a longer time to converge. If n is the population size, the convergence is expected after $n \log (n)$ function evaluations (Nadeen, 2000).

The use of today's new parallel computers not only provides more storage space but also allows the use of several processors to produce and evaluate more solutions in a smaller amount of time. By parallelizing the algorithm, it is possible to increase the population size, reduce the computational cost, and so to improve the performance of the genetic algorithms.

Probably the first attempt to map genetic algorithms to existing parallel computer architectures was made in 1981 (www.ecpp.ac.uk). But obviously today, with the emergence of new high performance computing (HPC), PGA is really a flourishing area. Researchers try to improve performance of GAs. The stake is to show that GAs

are one of the best optimization methods to be used with high performance computation.

The first attempt to parallelize genetic algorithms simply consists of global parallelization. This approach tries to explicitly parallelize the implicit parallel tasks of the 'sequential' genetic algorithm. The nature of the problems remains unchanged. The algorithm still manipulates a single population where each individual can mate with any others, but the breeding of the new children and/or their evaluation is now made in parallel. The basic idea is that different processors can create new individuals and compute their fitness in parallel almost without any communication between each other.

To start with, doing the evaluation of the population in parallel is something really simple to implement. Each processor is assigned a subset of individuals to be evaluated. For example, on a shared memory computer, individuals could be stored in shared memory, so that each processor can read the chromosomes assigned and can write back the result of the fitness computation. This method only supposes that the GA works with a generational update of the population. Of course, some synchronization is needed between generations.

Generally, most of the computational time in a genetic algorithm is spent calling the evaluation function. Time spent manipulating the chromosomes during the selection or recombination phase is usually negligible. By assigning to each processor a subset of individuals to evaluate, a speed-up proportional to the number of processors can be expected if there is a good load balancing between them. However, load balancing should not be a problem, as generally the time spent for the evolution of an individual does not really depend on the individual. A simple dynamic scheduling algorithm is usually enough to share the population between each processor equally.

On a distributed memory computer, the population can be stored in one 'master' processor responsible for sending the individuals to the other processors, 'the slaves'.

The master processor is also responsible for collecting the result of the evaluation. A drawback of this distributed memory implementation is that a bottleneck may occur when the slaves are idle while only the master is working. But a simple and good use of the master processor can improve the load balancing by distributing individuals dynamically to the slave processors when they finish their jobs.

A further step could consist in applying the genetic operators in parallel. In fact, the interaction inside the population only occurs during selection. The breeding, involving only two individuals to generate the offspring could easily be done simultaneously over *n/2* pairs of the individuals. But it is not that clear if it is worth doing so. Crossover is usually very simple and not so time consuming; the point is not that too much time will be lost during the communication, but that the time gain in the algorithm will be almost nothing compared to the effort produced to change the code.

This kind of global parallelization simply shows how easy it can be to transpose any genetic algorithm onto a parallel machine and how a speed-up sub-solution to the number of processors may be expected. As opposed to the methods introduced in the next two paragraphs, this parallelization is done without changing anything of the nature of the algorithm.

4.2.1. Coarse Grained PGAs - The Island Model

The second class of parallel GA is once again inspired by nature. The population is now divided into a few subpopulations, which are called "*demes*", and evolves separately on different processors. Exchange between subpopulations is possible via a migration operator. The term islands model is easily understandable; because the GA behaves as if the world was constituted of islands where populations evolve isolated from each other. On each island the population is free to converge toward different optima.

An individual can no longer breed with any other from the entire population, but only with individuals of the same island. Amazingly, even if this algorithm has been developed to be used on several processors, it is worth simulating it sequentially on one processor. It has been shown on a few problems that better results can be achieved using this model. This algorithm is able to give different sub-optimal solutions, and in many problems, it is an advantage if one needs to determine a kind of landscape in the search space to know where the good solutions are located. Another great advantage of the island model is that the population on each island can evolve with different rules. That can be used for multi-criteria optimization. On each island, selection can be made according to different fitness functions, representing different criteria. For example, it can be useful to have as many islands as the criteria, plus another central island where selection is done with a multi-criteria fitness function. The migration operator allows individuals to move between islands, and therefore, to mix the criteria.

In the literature, this model is sometimes also referred to as the coarse grained parallel GA. (In parallelism, the grain size refers to the ratio of time spent in computation and time spent in communication; when the ratio is high the processing is called coarse grained). Sometimes, the term "distributed" GA can be found, since they are usually implemented on distributed memory machines (www.epcc.ac.uk).

Technically there are three important features in the coarse grained PGA: the topology that defines connections between subpopulations, migration rate that controls how many individuals migrate, and migration intervals that affect how often the migration occurs. Even if a lot of work has been done to find optimal topology and migration parameters, here, intuition is still used more often than analysis with quite good results. Figure 4.1 shows comparison of Island model with stepping stone model in parallel genetic algorithms. Island model includes more migration roads than stepping stone model does. Therefore, it can be executed in a shorter time by considering much more alternative paths.

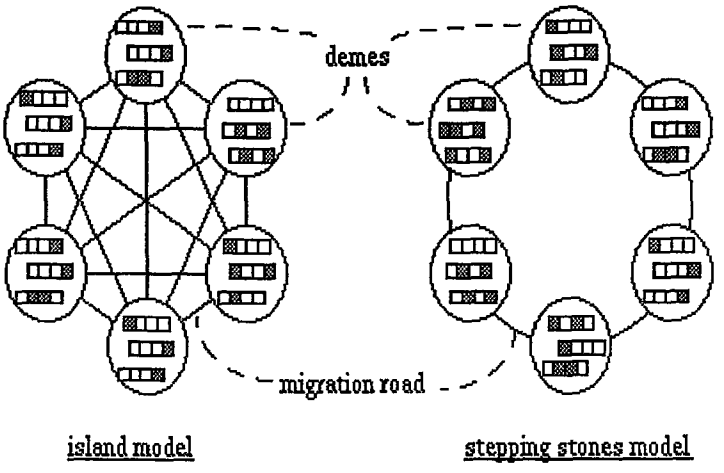


Figure 4.1. Island Model of Parallel Genetic Algorithm and Stepping Stones Model
Source: www.epcc.ac.uk

Many topologies can be defined to connect the demes, but the most common models are the island model and the stepping-stones model. In the basic island model, migration can occur between any subpopulations, whereas in the stepping stone demes are disposed on a ring and migration is restricted to neighboring deems. Works have shown that the topology of the space is not so important as long as it has high connectivity and small diameter to ensure adequate mixing as time proceeds.

Choosing the right time for migration and which individuals should migrate appears to be more complicated. Quite a lot of work is done on this subject, and problems come from the following dilemmas. It appears also that, immigration is a trigger for evolutionary changes. If migration occurs after each new generation, the algorithm is more or less equivalent to a sequential GA with a larger population. In practice, migration occurs either after a fixed number of iterations in each deem or at uniform periods of time. Migrants are usually selected randomly from the best individuals in the population and they replace the worst in the receiving deme. In fact, intuition is still mainly used to fix migration rate and migration intervals; there is absolutely nothing rigid; each personal cooking recipe may give good results (Ghrayeb, 2000).

4.2.2 Fine Grained PGAs

The development of massively parallel computers triggers a new approach of parallel genetic algorithms. To take advantage of new architectures with even a greater number of processors and less communication costs, fine-grained PGAs have been developed. The population is now partitioned into a large number of very small subpopulations. The limit (and maybe ideal) case is to have just one individual for every processing element available.

Basically, the population is mapped onto a connected processor graph, usually, one individual on each processor. (But it works also with more than one individual on each processor. In this case, it is preferable to choose a multiple of the number of processors for the population size). Mating is only possible between neighboring individuals, i.e. the individuals stored on neighboring processors. The selection is also done in a neighborhood of each individual and so depends only on the local information. A motivation behind local selection is biological. In nature there is no global selection, instead natural selection is a local phenomenon, taking place in an individual's local environment.

If this model is compared with the island model, each neighborhood can be considered as a different deme as given in Figure 4.2. But here, the demes overlap providing a way to disseminate good solutions across the entire population. Thus, the topology does not need to explicitly define migration roads and migration rate.

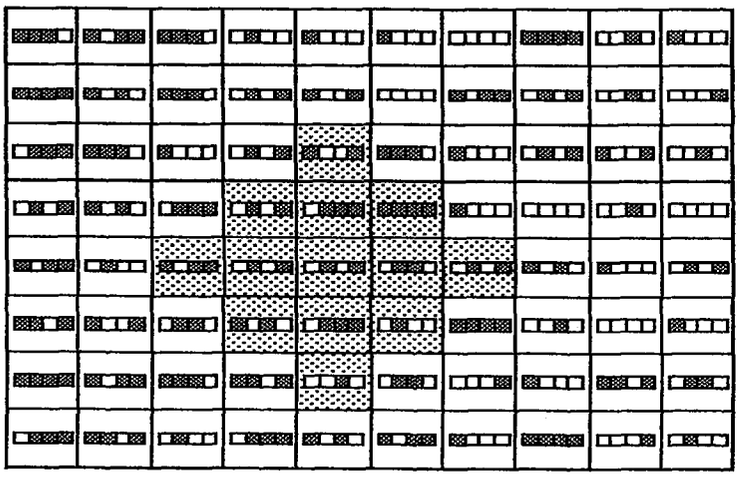


Figure 4.2. Representation Example of Fine Grained Parallel Genetic Algorithms. Source: www.epcc.ac.uk

It is common to place the population on a 2-D or 3-D grid because in many massively parallel computers the processing elements are connected using this topology. Consequently each individual has four neighbors. Experimentally, it seems that good results can be obtained using a topology with a medium diameter and neighborhoods not too large. Like the coarse grained models, it is worth trying to simulate this model even on a single processor to improve the results. Indeed, when the population is stored in a grid like this, after few generations, different optima could appear in different places on the grid.

To sum up, with parallelization of GA, all the different models proposed and all the new models one can imagine by mixing those ones, can demonstrate how well GA are adapted to parallel computation. In fact, too many implementations reported in the literature may even be confusing. For many researchers, the real great advantage of GA over other optimization techniques, like simulated annealing is their ability to be parallelized.

4.3. Advantages and Disadvantages of Genetic Algorithms

Genetic algorithms have many advantages when compared with the other heuristic techniques; the major advantages are presented below.

1. Generality

Since genetic algorithms work on different coding systems, only one general computer program may solve different decision problems when solution spaces are selected carefully.

2. Nonlinearity

Genetic algorithms do not need any assumption such as linearity, convexity, differentiability, thus GAs are nonlinear algorithms. GAs only require performance measures and furthermore these measures may not be defined by a mathematical expression.

3. Robustness

A Genetic algorithm with right encodings, parameter settings (population sizes, mutation and crossover rates) and operators presents acceptable and robust solutions.

4. Ease of Modification

For many problems, even minor modifications may be so difficult to apply to heuristics, but for GA heuristics, it is easy to change the model according to the variations of the original problem.

5. Parallel Nature

There is a great potential for implementing GAs in parallel and it will be increasingly common.

Genetic algorithms provide solutions to problems in different study fields. Genetic Algorithms have been most popular techniques to be used; especially 1990s were golden era of this method. Table 3.1 summarizes the problem types in which Genetic algorithms developed successful solutions.

Genetic algorithms can be used as an integrated model with the other search tecniques. In topology optimization of neural networks, and genetic algorithms are used to build the more appropriate architecture of the network, how many hidden layers and how many hidden nodes should be used, how the nodes should be interconnected (Kitano, 1990) and (Gruau, 1992).

Beside these advantages, genetic algorithms have some disadvantages. It is good to remember that the efficiency of a genetic algorithm is extremely dependent on consistent coding and relevant reproduction operators. Building a genetic algorithm, which performs no more than a random search, happens more often than one can expect. If the reproduction operators are just producing new random solutions without any concrete links to the ones selected from the last generation, the genetic algorithm is just doing nothing else than a random search. The model should fit to the GA terminology, otherwise the algorithm cannot provide even feasible solutions. The other disadvantage is about the heuristic structure. GAs search the population under the defined heuristic procedure, because they do not guarantee an optimal solution. If the operators of GA are not defined clearly, the algorithm may stop at a local optimum, but this situation can be controlled by keeping the diversity of the population at a specific level.

Table 4.1. Successful Applications of Genetic Algorithms

BIN-PACKING	GRAPH RELATED	LOCATION	SCHEDULING
Falkenauer and Delchambre (1995) Reeves (1996) Smith (1985)	Höhn and Reeves (1996) Jones and Beltramo (1991) Von Laszewski (1992) VonLaszewskiand Mühienbein (1991) Pirkul and Rolland (1994)	Brown, Huntley, and Spilane (1989) Conway and Venkataramanan (1994) George (1996) Houk, Joines, and Kay (1996) Huntley and Brown (1991) Palmer and Kersenbaum (1995) Tam (1992) Tate and Smith (1995)	Cartwright and Mott (1991) Clevelend and Smith (1989) Davis (1991) Della Croce, Tadei, and Volta (1995) Dorndorf and Pesh (1995) Fang, Ross, and Corne (1993) Kobayashi,Ono,andYamamura (1995) Reeves (1995) Starkweather et al. (1991) Syswerda and Palmucci (1991) Yamada and Nakano (1996)
SET RELATED	**TRAVELING SALESMAN**	**STEINER TREE**	
Aggarwal, Orlin, and Tai (1997) Al-Sultan,Hussain, and Nizami(1996) Beassey and Chu (1996) Levine (1996) Liepins ad Potter (1991) Richardson et al. (1989)	Freisleben and Merz (1996) Homaifar, Guan, and Liepins (1993) Jog, Sub, and Van Gucht (1991) Mathias and Whithley (1992) Mühlenbein (1991) Oliver,Smith, and Holland (1987) Prinetho, Rubeudengo, and Sonza Reonda (1993) Ulder et al. (1991) Whitley, Starkweather, and Shaner (1991)	Esbensen (1995) Hesser, Manner, and Stucky (1989) Juistrom (1993) Kapsalis, Smith, and Raywand-S (1993)	**TRANSPORT SCHEDULING** Blanton and Wainwright (1991) Thangiah, Vinayamoorthy, and Gubby (1993) Wren and Wren (1995)

Source: Genetic Algorithms for Operations Researcher, Reeves, Colin R.; INFORMS Journal on Computing; 1997,vol9/03,p 242.

4.4. GA based Scheduling

A complete schedule is derived from a time dependent preference list for each machine. A "time period" is associated with each preference list, which contains some permutation of contracts (groups of jobs), plus the elements "wait" and "idle". This list is interpreted, as showing which contract the machine should prefer to execute at the associated time or whether it should wait or stand idle. The crossover operator exchanges preference lists for selected machines.

Several GA based approaches to tabu search problems found the introduction of domain knowledge to be most important source of high performance operators and led to clues for more effective representations and operators for scheduling problems. Applicability and effectiveness of a variety of non-standard representations and operators was investigated and introduced from GA work on tabu search problems (Glover,1993).

It is possible to achieve balanced optimization in a situation where GA's only generate partial schedules, which are then evaluated based on the characteristics of machine environment. Although job shop scheduling problems can be reduced to the traveling salesman problem, some restrictions and simplifications need to be made. The traveling salesman problem-like approaches have their limitations when applied to more complicated job shop scheduling problems. Some researchers have abandoned the traveling salesman problem based approaches and developed more realistic representations and more complicated operators for Job shop scheduling problems. Binary representation has been used to select the disjunctive arcs and conventional operators to generate new strings. Here, the schedule builder not only produces schedules from feasible strings, but also applies a complicated repair procedure to find feasible schedules from infeasible strings. A treatment called Forcing replaces an infeasible string with a similar, but feasible string. A conventional GA can effectively solve scheduling problems; first, if the schedule builder is well designed. Second, the replacement of infeasible strings with feasible

ones can help the population converge quickly, but too many replacements may cause premature convergence.

The results show that a representation comprising complete information ends up with a good solution, but the slower convergence due to the large search space prevents the applicability on real world scheduling problems (O'Reilly, 1995).

A general description of scheduling problems can be simply expressed as a question of how to allocate efficiently a set of resources (machines, people, rooms, facilities) to carry out a set of predefined tasks. The suitable solutions must respect a number of hard constraints. The more obvious constraints are that resources are limited and that any resource cannot be used to perform more than one operation at the same time. Other constraints can be that the different tasks may have to be accomplished in a specific order. The suitable solution must also minimize the cost. The cost can be defined as the earliest overall completion time or amount of "idle time" for each resource. Generally, scheduling engenders large and difficult combinatorial problems. Most of them are known to be non-deterministic polynomial, i.e. they are not currently solvable using a deterministic polynomial time algorithm. Recently, a lot of work has been done to apply genetic procedure to this kind of difficult problems (Lim, 1997). It really seems that genetic algorithms are able to produce successful and quick solutions of hard scheduling problems, often real problems.

Frequently, scheduling problems are solved using a simplified model studied as a mapping problem in a parallel computation. The problem of assigning each process a processor was done using a simple mapping function. Two contradictory mapping criteria have been considered: to minimize the sum of the total communication cost between processors, and to minimize the load imbalance across the system. The cost function was a weighted sum of the total communication cost and the loads variance of the different processors. A simple genetic representation was used corresponding to a simple model (Nolfi et al, 1994). As many genes as processes composed the genome, each gene representing the processor assigned to the process. A parallel

genetic algorithm has been implemented by using a fine-grained model, and compared with two other algorithms, a hill climbing and a simulated annealing. The results show that the hill-climbing gives worse quality solutions compared to simulated annealing but they are faster. Genetic algorithms give comparable solutions as simulated annealing, but with a search time, which is of the same order of magnitude than the hill climbing. They concluded that one of the main advantages of genetic algorithms was that they are intrinsically parallel (www.epcc.ac.uk).

Scheduling is a specific planning problem involving the need to assign time slots to a set of events, subject to constraints on these assignments. The most common and important such constraint in scheduling problems is an edge constraint between two events, so called due to the similarity of the simple scheduling problems with vertex coloring problems in graph theory. An edge constraint simply states that a given pair of events *e1* and *e2* must not overlap in time. It may be because, for example, the same teacher gives two classes, or because one or more students must attend both of them. A related kind of constraint is what is called an event-spread constraint. This is an extra requirement on the relative times assigned to pairs of edge constraint event, like consecutive classes.

Scheduling may also involve the assignment of rooms or places as well as times; this brings constraints arising from room capacities and/or travel-time between locations. Solving such a problem involves at least finding an acceptable timetable - one in which all hard (obligatory) constraints are satisfied, and for which its violations of soft (required but not necessary for an acceptable solution) constraints are suffered. At best, solving such a timetable problem involves finding or coming close to an optimal minimal number of constraint number violations (Ross, 1994). Often, real scheduling problems seem not very hard if the edge constraints should be taken into account. This fact explains why a lot of timetables are done by hand. The trouble is the inclusion of event-spread constraints, along with all sort of soft, although strongly desired constraints (such as preferred ordering for events, teachers' preferred classes, morning off, and so on) result in a very difficult multi-objective

optimization problem, which requires considerable computational effort for its solution.

A simple approach to resolve scheduling problems is to use a *direct* representation. If *e* is the number of events, the direct representation consists of a chromosome of length *e*, where each gene typically corresponds to the possible start times (or time slot) of an event. The *i*th allele thus represents the time at which event *i* begins. In a problem involving room and teachers, for example, a simple direct interpretation of *"(abc)(def)..."* might be event *e1* starts at time *a* in place *b* involving teacher *c*, and event *e2* starts at time *d* in place *e*, involving teacher *f*. The fitness function commonly used in this case is a weighted sum of constraint violations; a candidate solution is tested against a long list of constraints, and the penalties are assigned for each violation according to its importance. This way, evolution results in lower and lower weighted penalty scores until at least timetables suffering no hard constraint violations appear, and (soon after) timetables appear which suffer few or no violations of soft constraints (Corne, 1994). This direct representation and successfully solved some real examination scheduling problems arise from MSc examinations at the University Of Edinburgh Department Of Artificial Intelligence, and from Kingston University in London. The results obtained were clearly better than the timetable produced by a human expert. Its also shown that similar results for a collection of real lecture scheduling can be obtained using direct representation (Ross, 1994).

A second approach to resolve scheduling problems consists in using an *implicit* representation. With an implicit representation the genes do not directly refer to the time slot of a given event, but to the way this event should be included in the timetable. An implicit representation relies on the use of a heuristic to build the timetable (a schedule builder). The schedule builder takes events in a specific order and successively assigns them to a time-slot that satisfies all the hard constraints (if any), in the partially build timetable. The schedule builder is thus supposed to only generate valid timetables, or at least timetables that violate only few hard constraints. So, the optimization is done in a reduced search space, without bothering with an invalid solution. The task of the genetic algorithm simply is to find the best order in

which the event should be taken into consideration for the heuristic. The chromosomes are interpreted as permutations. The schedule builder, unlike the genetic algorithm, completely understands the details of the scheduling task. The question of "how smart the schedule builder should be" depends on how much faith one places in the genetic algorithm portion of the system. If the schedule builder is not very clever, it will seldom construct very good schedules.

The direct representation seems to have difficulties in dealing with large size problems. But because of its simplicity it is easy to build a specific mutation operator that tries to reduce constraint violations and then improve the quality of the result. On the other hand, even if the implicit representation seems better at solving large and difficult problems because it searches in a far fitter sub-space, too little work has been done on this representation to be totally convinced of its real efficiency (Mitchell, 1996).

Genetic Algorithms are one of the most common local search algorithms that are used with the other search techniques as hybrid methods. For instance, a column generation technique was used with a genetic algorithm for parallel machine scheduling to minimize makespan and total cost of changeovers. In this study, roulette wheel method was used for chromosome selection to reproduce a new generation, and PMX type crossover operator was chosen to obtain sequence of jobs considering the objective given above (Figielska, 1999).

Another Genetic Algorithm was developed for job shop scheduling problem with unrelated parallel machines. Precedence constraints between operations of jobs were considered. Indirect domain – independent representation was applied, which means the coding, does not include the whole solution. Permutation based crossover and random mutation operators were used for reproduction. A specific heuristic function was developed for machine selection and shortest process time was chosen to determine the next operation. This algorithm gives the best results when the population size is 50 (Ghedjati, 1999). Another genetic algorithm was developed for single machine scheduling using similar heuristics (Liu & Tang, 1999).

Most GA –based research only studied the static scheduling problems with the makespan objective. In dynamic scheduling problems, jobs can arrive at some known or unknown future times. Further, the importance of each job can be different and the objective can be more complex. From a practical view, a large number of priority rule approaches have been proposed and tested for dynamic problems. While priority rules are computationally efficient and are useful for finding a reasonably good solution, prioritizing the jobs based on only a few job or machine parameters restricts the search capability. The costs of the computations are also important for today's problems.

Many GA-based approaches need specific operators to ensure the validity of the solution or to integrate the domain knowledge. Although the specific operators are difficult to design, if the problem-specific knowledge is successfully applied into the operators, the GA can work more effectively on the particular problem. Temporary relationships among all operations in a schedule are important. Working on the chromosome level usually focuses on only a small part of the schedule and overlooks the change of the temporary relationships in the whole schedule.

Some problem-specific indirect representations suffer from the problem of false competition. In some indirect representations, such as prioritization of scheduling rules, it is difficult to encode the schedule back to the chromosome.

When rescheduling problems are considered, a more elaborate method is necessary to deal with the trade-off between schedule quality and processing time.

Discarding the old population and then constructing the new schedule from scratch result in another rescheduling approach. Consider the final population in the last time period before new jobs arrive. Typically only a few operations are removed from the last problem to generate the new problem, only a small fraction of the information.

CHAPTER FIVE
PROPOSED APPROACH

5.1. Introduction

In this chapter, a scheduling model is developed using genetic algorithms, which is one of the popular meta-heuristic techniques. As explained in chapter 3, Meta-heuristics (or modern heuristic techniques) present nearly optimal solutions for real world problems and can be used for short term planning problems.

Genetic algorithms have many advantages presented in chapter 4 to be used as a tool for scheduling (Figielska, 1999). (Ghedjati,1999), (Sakawa, 1999), and (Kimms, 1999) present successful examples of genetic algorithms used for scheduling.

5.2. The Environment of Implementation

5.2.1. Company Information

PMS LTD. STI., a rotational molding company, was founded in İzmir in 1993. At the beginning, it served food and textile industry and produced such products as fermentation tanks, brine silos, different types of carts, and stacking boxes. Today, it is a leader in the blow molding, injection molding, and rotational molding industries. The major products of the company are plastic containers in various volumes and shapes for all industries such as chemical, textile and agriculture.

PMS has a production capacity shown in Table 5.1.

Table 5.1. Production Capacity of PMS

PRODUCT TYPE	CAPACITY (TON/YEAR)
Rotational molding	2000
Blow molding	1500
Injection molding	250

PMS has two computer controlled rotational molding machines and two shuttle type home made rotational molding machines. In PMS facilities, there is a metal workshop and pattern house to make molds and they are also sold. The firm has its own Grinding, Recycling, and Compounding and Granulating units.

All the molds have been designed with Solid Works, which is a software packet for the design of major products of the company.

Products of the company are:

- Vertical, horizontal, and silo type tanks to be used for keeping water, acids, food concentrate, etc.
- Marine products (boat, docks, brackets for offshore cages)
- Pallets
- Barrels
- Waste bins

The production technology depends on rotational molding. The production area is divided into three main departments; raw material setup, mold setup, and rotation and construction processes. According to the customer demand, in raw material setup area, white polietilen granules are converted to powder with desired color and handled to the molding area. At the same time, the required mold is set up, then handled to the rotation area and mounted to the rotation machine. The relationships in this shop floor have been represented in Figure 5.1.

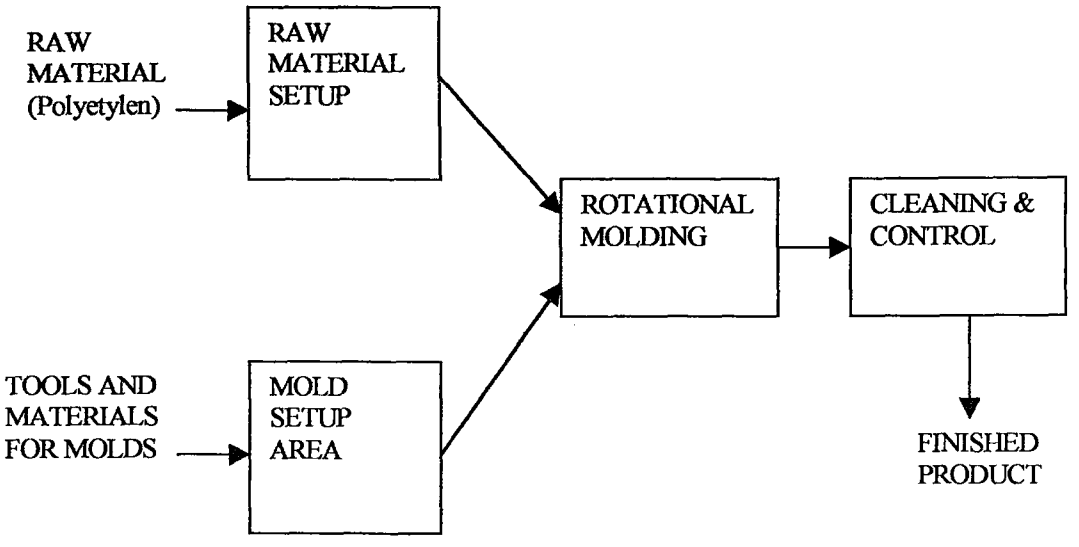


Figure 5.1. Process Flow of Shop Floor in the Company.

5.2.2. Definition of the Problem

Scheduling of the parts based on customer orders is one of the most important problems in the shop floor. All parts have very long process and setup times. Hence, short-term plans should be made effectively to meet the due dates. Main processes are carried out in the rotational molding area, and this is the department that contributes to the manufacturing lead-time mainly. More than one part can be processed in a cycle if volume and moment conditions are met. Setup times are too long and to be minimized. The scheduling problem is regarded as the grouping and sequencing the products to meet these objectives.

The problem is about one of the rotational molding machines called DKF60. There are six fixtures on this machine to place products. Thus, at most 6 products can be processed in one cycle according to the volume and total available area constraints on the machine. Therefore, in each cycle, one load of parts is selected to be produced.

When the parts are set on the machine, rotation process is started with the related torque. The torque of the machine is very important for the quality of the products. Hence, the volume of the products that are set up on the machine must not exceed 2000 liters. Setup times are too high for each cycle, and the machine utilization is also important. To provide minimum setup time and maximum volume on the machine, product groups and their sequences should be selected carefully. Hence, the problem is summarized as follows:

"WHICH GROUP OF PRODUCTS SHOULD BE PROCESSED IN WHICH SEQUENCE ON THE MACHINE?"

This problem is a single machine-scheduling problem. All products processed on this machine have the same process times, and these times are nearly the same for each cycle. It is assumed that process and setup times are deterministic.

In summary, there are two constraints on the machine to be considered:

- The total volume of the group to be set up for a cycle must not exceed 2000 liters.
- Since there are 6 fixtures on the machine, one load can include 6 products at most in a cycle.

The decision variables are:

- Volume of a product, and total volume of a load (V_i *and* T_j *respectively).*
- Number of products within the group (m).
- Setup time of a product (sp_i), and total setup time of the group (S_j).
- Order size of the product (O_i).

The objective is to determine the groups and their sequences in a schedule that provides minimum setup times and maximum machine utilization.

Although the problem is single machine scheduling, the solution space contains large number of combinations to search. For example, if there are 10 products, there are $\sum_{i=1}^{6} C(10,i)$ combinations only for load creation. There is also large number of permutations between the loads created. Therefore, it is impossible to solve this problem by considering all constraints for the combinations and permutations in an acceptable time interval. Under these conditions, it is more appropriate to use a meta-heuristic method. In this study, a model is proposed including a heuristic function that is developed for group selection to obtain optimum or near optimum sequencing plan.

5.3. Characteristics of the Proposed Approach

The model has been developed for dynamic and deterministic single machine scheduling problems and can be used under some assumptions:

1. Operation times are deterministic.
2. Parts on the machine cannot be removed until the operation has finished, i.e. this is a non-preemptive model.
3. Scrap and machine breakdowns are neglected.

5.3.1. The Priority Rule

Group selection and sequencing are constructed according to a priority rule based on these objectives, so the proposed method employs a priority index considering the relationship between setup and processing times. It also considers the variation of order sizes within a particular group. This priority index is used for the selection of the next best group. This selection process is based on a genetic algorithm. The following gives the calculation of the priority index.

For each group there are three variables to optimize:

- Maximize the total volumes of groups for machine utilization.
- Maximize the ratio of "Process time / setup time" to maintain maximum process time per setup.

- Minimize variation of the order sizes within the group, because less variation provides less number of setups in a particular group of parts. The variation is measured by using the coefficient of variation of the group.

To obtain a single priority index to be minimized, the following steps have been implemented:

Step1. Calculate the volume of each load:

$$T_j = \sum_{i=1}^{m} V_i$$

where

V_i : *volume of product i.*

T_j : *total volume of the load j.*

m : *number of products within a load*

Total volumes are ranked by giving minimum rank to the load with highest volume. The less the rank is the higher the priority is. The volume rank of load j is denoted by (RV_j).

Step2. Calculate the Process time - setup time ratio, R_j, of each group.

$$R_j = P_j / S_j$$

$$S_j = \sum_{i=1}^{m} sp_i \quad \text{and} \quad P_j = \sum_{i=1}^{m} pt_i$$

where

sp_i: *setup time of product i.*

pt_i: *process time of product i.*

S_j= *total setup time of group j.*

P_j= *total process time of group j.*

These values are ranked by giving minimum rank to the group with highest rate The less the rank is the higher the priority is. The ratio rank of group j is denoted by (RP_j).

Step3. Calculate the variation within each group:

$$CV_j = \left(\frac{S_{oj}}{M_{oj}}\right) \qquad M_{oj} : \frac{\sum_{i=1}^{m} O_i}{m} \qquad S_{oj} = \sqrt{\frac{\sum_{i=1}^{m} (O_i - M_{oj})^2}{m}}$$

Where

O_i: order size of product i.

S_{oj}: Standard deviation of the order sizes within the group j.

M_{oj}: Mean of the order sizes within the group j.

These values are ranked by giving the minimum rank to the group with lowest coefficient of variation value. The less the rank is the higher the priority is. The variation rank of group *j* is denoted by (RC_j).

Step4. Normalize each rank of the group by its own average value including all groups (or members in GA terminology) of the current population. There are three main factors to be used together in the evaluation function. Rank values do not change in the same interval of equal length, and the unit of each factor is different from each other, thus, rank values may cause problems to analyze the factors effectively. Normalization should be implemented in order to define relationships among these ranks by eliminating rank variations within factors. For this purpose, the most applied normalization procedure is to divide each rank value by its mean. Normalization procedure has been formulated as follows.

$$Norm\_RV_j = \frac{RV_j}{AVG\_RV} \qquad Norm\_RC_j = \frac{RC_j}{AVG\_RC} \qquad Norm\_RP_j = \frac{RP_j}{AVG\_RP}$$

Where

RV_j= volume rank of group j.

RC_j= coefficient of variation rank of group j.

RP_j= Pj/Sj ratio rank of group j.

$Norm\_RV_j$= normalized volume rank of group j.

$Norm\_RP_j$= normalized Pj/Sj ratio rank of group j.

$Norm\_RC_j$= normalized coefficient of variation rank of group j.

AVG_RV= Mean volume rank for current population

AVG_RP= Mean Pj/Sj ratio rank for current population

AVG_RC= Mean coefficient of variation rank for current population

Step5. Average of normalized ranked values for each group is calculated

$G_j = \frac{(Norm\_RC_j + Norm\_RV_j + Norm\_RP_j)}{3}$, which indicates Priority factor of group j in the current population.

Therefore, a single priority factor is obtained for each group of a particular phase. Since set up time and machine utilization variations depend upon the volume of the product, all possible volumes have been chosen for implementation. Therefore, it is more appropriate to obtain a unique equation that gives a priority index, and apply it to future plans. For this purpose, discriminant analysis has been implemented on the average of normalized rank values to obtain the effects of each measure (volume, process time/setup, and coefficient of variation within the group) presented as G_j above by using SPSS 11.0 software. Syntax of this analysis is presented in *Appendix C.*

Discriminant analysis is a useful tool for situations to build a predictive model of the group membership based on observed characteristics of each situation. The procedure generates a discriminant function (or, for more than two groups, a set of discriminant functions) based on linear combinations of the predictor variables that provide the best discrimination between the groups. The functions are generated from a sample of cases for which group membership is known; the functions can then be applied to new cases with measurements for the predictor variables. Discriminant analysis allows estimating coefficients of the linear discriminant function, which looks like the right-hand side of a multiple linear regression equation.

The weight of each objective has then been defined by this analysis. The results of the discriminant analysis are shown in Tables 5.2, 5.3, and 5.4, 5.5, 5.6:

Table 5.2. Analysis Case Processing Summary

Unweighted Cases		N	Percent
Valid		6576	100.0
Excluded	Missing or out-of-range group codes	0	0.0
	At least one missing discriminating variable	0	0.0
	Both missing or out-of-range group codes and at least one missing discriminating variable	0	0.0
	Total	0	0.0
Total		6576	100.0

Table 5.3. Group Statistics

		Valid N	(list wise)
NORM_ORT		Unweighted	Weighted
000000000000	Norm_PRO_SET	5924	5924.000
	Norm_COEFVAR	5924	5924.000
	Norm_VT	5924	5924.000
1.000000000000	Norm_PRO_SET	652	652.000
	Norm_COEFVAR	652	652.000
	Norm_VT	652	652.000
Total	Norm_PRO_SET	6576	6576.000
	Norm_COEFVAR	6576	6576.000
	Norm_VT	6576	6576.000

NORM_ORT: normalized rank average of the group
Norm_COEFVAR: normalized rank coefficient of variation of the group
Norm_VT: normalized volume rank of the group
Norm_PRO_SET: normalized (process time/setup time) rank of the group

Table 5.4. Eigenvalues

Function	Eigenvalue	% of Variance	Cum.(%)	Can.Corr.
1	0.595[a]	100.0	100.0	0.611

First 1 canonical discriminant functions were used in the analysis.

Table 5.5. Wilk's Lambda

Test of Function(s)	Wilks' Lambda	Chi-square	df	Sig.
1	0.627	3069.481	3	.000

Table 5.6. Standardized Canonical Discriminant Function Coefficients

Variables	**Function**
R_j	-0.730
CV_j	0.953
T_j	-0.989

Finally, priority index for each group is obtained through the following equation which was calculated by using normalized rank values:

$$I_j = -0.730R_j + 0.953CV_j - 0.989T_j$$

Thus, the normalized ranks are calculated only once, and the equation above can be used at any time in the planning period. The discriminant function shows that each additional unit of *Pj/Sj ratio* decreases the index by *0.730,* each additional unit of coefficient of variation for order sizes increases the index by *0.953*, and each additional unit of volume decreases the index by *0.989*. This discriminant function is used as the "heuristic function of the model", or in other words, it is the evaluation function developed for the proposed model. So these values are used as weights of ratio, volume, and order size variations.

5.4. The Algorithm of the Model

The algorithm of the model includes three modules given below:

1. Creating the initial population (solution space).
2. Evaluating each product group (subpopulation) based on the heuristic function.
3. Implementation of dynamic genetic algorithm for each phase to construct the appropriate sequencing for the schedule.

The heuristic function presented in section 5.3 is implemented for the selection of the best group in each phase, and a genetic algorithm is used for the sequencing process as a local search procedure.

Representation

Groups (or Chromosomes), which are the candidates for the optimum solution in the genetic algorithm, are considered as strings with 6 locuses. Each locus may include one of the products in the plan or is zero. For instance, one of the groups is shown below to explain the representation code of the algorithm:

P1	P1	P2	P15	0	0

Calculations are carried out for the volume, order size, process time, and setup time of the product in each locus within the group or chromosome.

Initial Population

A genetic algorithm procedure starts with creating an initial population. In this work, the initial population consists of all combinations of the product loads, i.e., chromosomes, including at most 6 products that meet the volume constraints. There exist 6576 loads in the initial population. Since all the data require about 135 pages to be shown, only 500 of them were chosen to show to the data pattern as presented in *Appendix A*.

Evaluation Function

The genetic algorithm then exploits a heuristic function. Most of genetic algorithm approaches include only one objective and consider relative frequency of the objective function (Mitchell, 1996). In this study, the heuristic function defined is based on a priority rule for the evaluation function, which is obtained after discriminant analysis of the three measures (volume, process time/setup time, coefficient of variation of a product group):

$$I_j = -0.730R_j + 0.953CV_j - 0.989T_j$$

This function has been obtained by using the values, which are normalized by average values for the first time. After discriminant analysis, a unique function has been obtained for all products, and this provides time, hardware and software advantages.

Selection Process

The selection of the chromosomes depends on the evaluation function (priority index) in each phase of the algorithm. The best group, i.e., the minimum function value, is then selected.

Cross-over Type

As mentioned in chapter 3, there are several types of cross-over processes. In this study, population based crossover is implemented on each phase of the sequencing process. Each setup can be considered as a different sequencing phase so that the best product group (or chromosome) can be selected for the optimum sequence. The algorithm starts with the group that has the minimum value. When one of the products is finished, it is removed from the machine. It is then decided which group will be the next through the crossover process below:

1. Crossover point(s) are determined based on the blank position in group (indicates locuses in the chromosome in GA literature).
2. The groups exchange their products based on the available locuses in the group. New children are then born. Thus, for each phase, the crossover point varies.
3. The new children are tested whether they meet the volume constraint, and the children who can be a member in the solution are added to the population, while the others are disposed. After this process, the new generation goes on the sequencing process. Following example shows the two-point crossover process, parent-1 and parent-2 exchanges its genes in locus-1 and locus-3 :

Parent-1:	0	P5	0	P5	P10	P15
Parent-2:	P3	P5	P6	P10	P14	P15
Child-1:	P3	P5	P6	P5	P10	P15
Child-2:	0	P5	0	P10	P14	P15

Mutation

If the same children are born in the last generation, mutation is implemented. Duplicated children are then disposed from the generation. The existence of the duplicated children can be tested any time, so there is no need to use any probability function for the mutation process.

Termination criteria

The algorithm is halted when all the products ordered in the planning period are completed.

Now the algorithm can be summarized as follows:

Step 1. Initialize the population, which indicates a feasible solution set.

a) Form all possible combinations of the number of products by 6-part-load for the initial population.. $\sum_{i=1}^{6} C(r,i)$ Where r is the number of products in schedule.

b) Revise the population by eliminating the combinations, which violate the volume constraint. Recall that this constraint indicates that maximum volume loaded on a machine cannot exceed 2000 liters.

Step 2. Calculate the heuristic function, which will be a beginning point of the schedule and used as a selection criteria as given below.

$$I_j = -0.730R_j + 0.953CV_j - 0.989T_j$$

Step 3. Select the group (chromosome) with the minimum I_j value.

Step 4. Production of the group of parts.

a) Produce the group until one of the product's order is finished.

b) To remove the finished parts from the chromosome, take the group as parent_1 and select a new group as parent_2. Using the crossover operator, exchange the products or genes except for the genes whose orders are not finished yet. After all crossover process is finished, add the new children to the population, which meet the volume constraint and set the population as the new generation.

Step 5. If there exists a product that has not been produced yet, go to step 2.

A flow chart of the algorithm is shown in Figure 5.2.

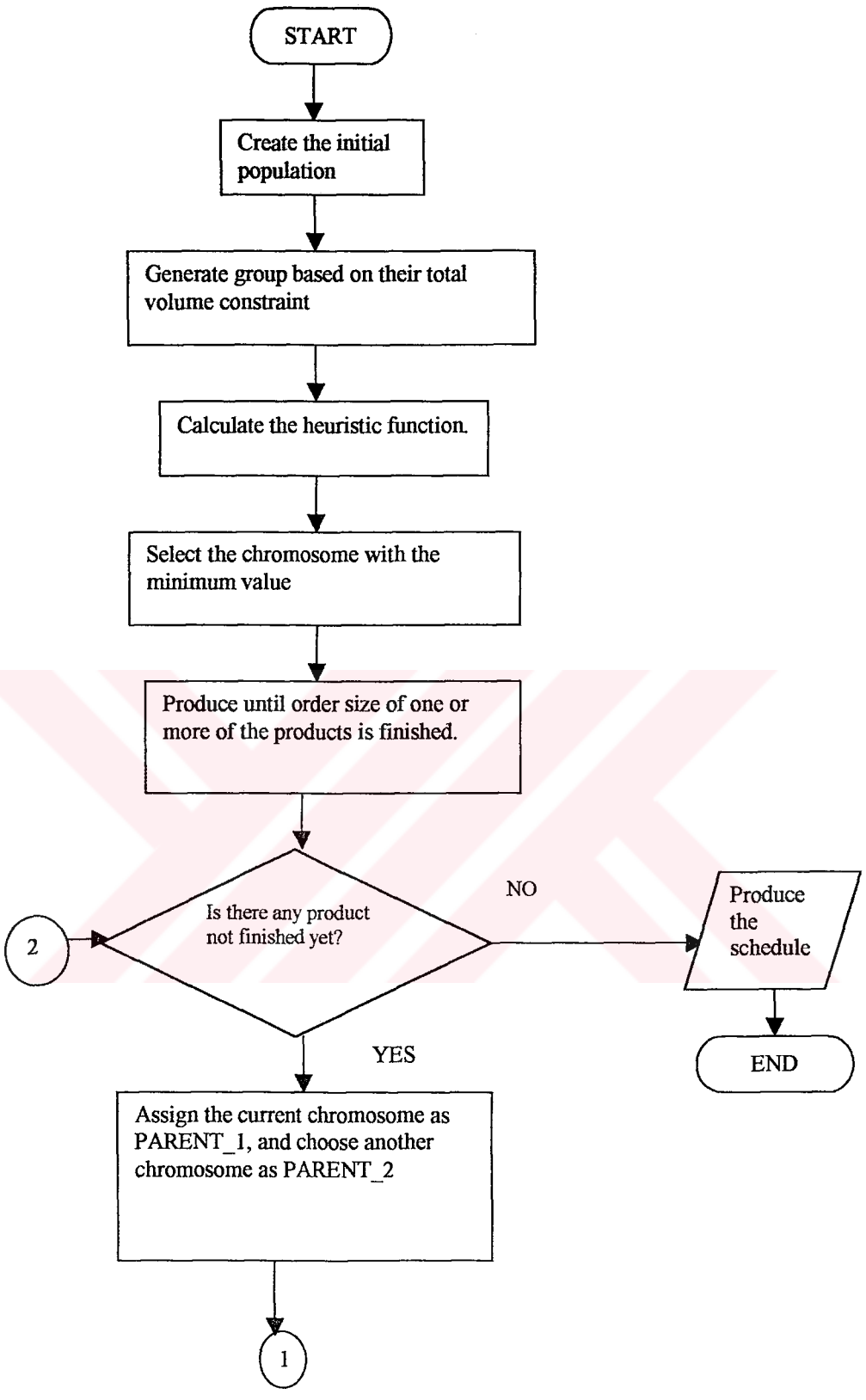


Figure 5.2. Flow Diagram of the Proposed Method

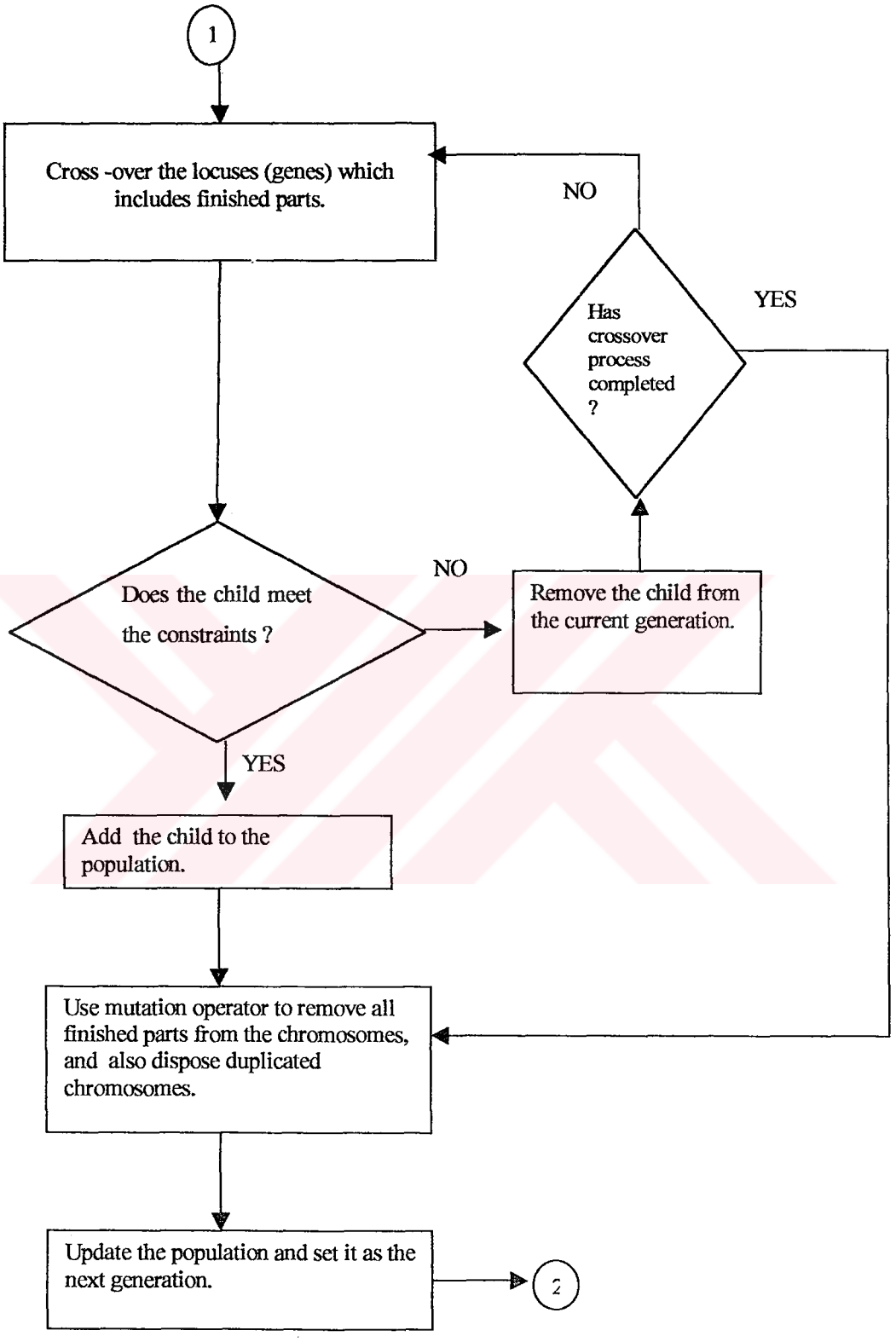


Figure 5.2. Flow Diagram of the Proposed Method (continued).

5.5. An Example for the Model

5.5.1 Products

PMS produces about 200 different products by using the machine (DKF60), but not all of them are ordered frequently. Hence, most frequent 15 products have been chosen for the model. These parts are listed in Table 5.7.

Table 5.7. Product List for Implementation

PR NO	Product ID	Definition	Volume (lt)	Setup Time (Min)	Oven (Min)	Cooling (Min)	Order Quantity (Unit)
P1	DD1-01000	Vertical tank	1000	33	25	30	15
P2	DY1-00160	Horizontal tank	160	23	25	30	50
P3	DY1-00250	Horizontal tank	250	23	25	30	100
P4	DY1-00500	Horizontal tank	500	28	25	30	75
P5	DY1-00750	Horizontal tank	750	30	25	30	200
P6	DY1-00700	Horizontal tank	700	30	25	30	150
P7	DY1-01000	Horizontal tank	1000	33	25	30	300
P8	DY1-01501	Horizontal tank	1500	32	25	30	120
P9	DY1-02000	Horizontal tank	2000	35	25	30	90
P10	SD5-00150	Horizontal silo	150	26	25	30	85
P11	TA1-00507	Textile car	500	28	25	30	35
P12	TA1-00544	Textile car	540	28	25	30	50
P13	TA1-00588	Textile car	580	28	25	30	45
P14	TK1-00069	Textile car	60	23	25	30	65
P15	TK1-00091	Textile car	90	23	25	30	80

Process times are assumed to be 55 minutes (the sum of oven and cooling) on average per cycle, and for each product, average order size and average set up time was considered. Operations for each cycle on the machine are given below:

1. Each mold is set on available fixture on the machine.
2. Raw material of each product is placed on each mold.
3. Rotation process is implemented.
4. After the rotation process, cooling process is started.

5.5.2.The Results and Evaluation

These fifteen parts are added to the record sets of the computer program developed, and at the end, the following schedule is obtained in Table 5.8 and 5.9 using the algorithm defined in section 5.4. Based on Table 5.8, the parts should be finished in the following sequence:

P1-P7-P9-P12-P6-P5-P4-P8-P13-P11-P15-P3-P2-P10-P14

Table 5.8. Solution Table

F1	F2	F3	F4	F5	F6	Vt	proc_time	Setup	Indices	ID
				P1		2000	946	66	-2,17144	54135
					P7	1000	16830	0	-1225,637	54255
					P9	2000	4985	35	-2,081243	54257
					P12	1990	2838	88	-2,058814	53860
				P6		1450	5500	0	-603,6838	53860
					P5	750	2750	0	-201,4918	54253
				P4		2000	4240	60	-2,057989	54175
					P8	1500	2530	0	-182,2315	54175
			P13	P13		2000	1395	130	-2,008229	45978
				P11		840	770	0	-314,0001	53754
					P15	340	2530	0	-261,384	52053
	P3					250	1100	0	-80,54725	46020
P2	P2	P2	P2	P2		950	691	141	-0,9624622	11637
					P10	150	4180	0	-301,3463	41829
P14	P14	P14	P14	P14	P14	360	743	138	-0,3591928	38754

Table 5.9. Production Schedule

Sequence no	F1	F2	F3	F4	F5	F6	ID
1	0	0	0	0	P1	P7	54135
2	0	0	0	0	0	P7	54255
3	0	0	0	0	0	P9	54257
4	0	0	0	P5	P6	P12	53860
5	0	0	0	P5	P6	0	53860
6	0	0	0	0	0	P5	54253
7	0	0	0	0	P4	P8	54175
8	0	0	0	0	0	P8	54175
9	0	P3	P11	P13	P13	P15	45978
10	0	0	0	P3	P11	P15	53754
11	0	0	P3	0	0	P15	52053
12	0	P3	0	0	0	0	46020
13	P2	P2	P2	P2	P2	P10	11637
14	0	0	0	0	0	P10	41829
15	P14	P14	P14	P14	P14	P14	38754

These results show that the groups should be processed by the sequence given in the production schedule in Table 5.9 according to the fixtures (F1...F6) on the machine. P1 and P7 are produced until P1 has finished, denoted by "sequence 1". Then P7 will continue alone until it has been completed. The third group is processed until P9 has finished. Then the forth group is set up and processed until P12 has finished. Then P6 and P5 are completed respectively. The groups follow the sequence in Table 5.8 in the same way with respect to the schedule in Table 5.9.

As presented in Table 5.8 and 5.9, the algorithm chooses the chromosomes (or product groups) with highest total volume by sequencing the groups including the least setup times. Figures 5.3, 5.4 and 5.5 show that the measures go worse for the last chromosomes in the sequence, thus the remained groups have less volume sizes.

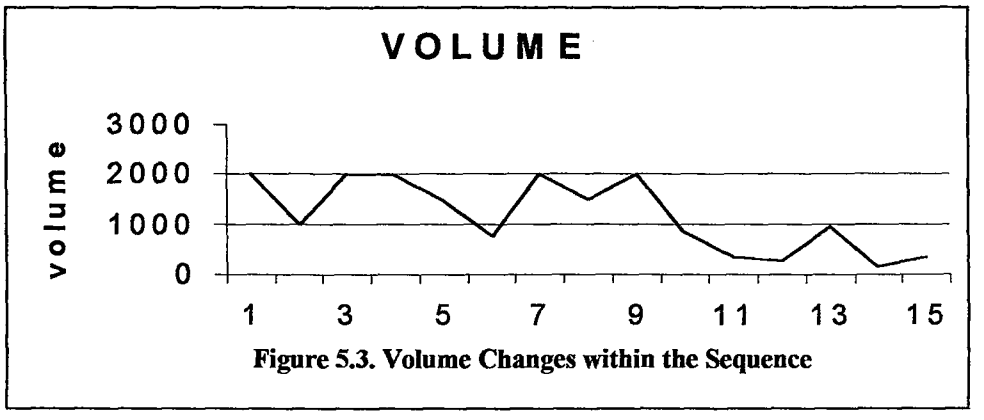


Figure 5.3. Volume Changes within the Sequence

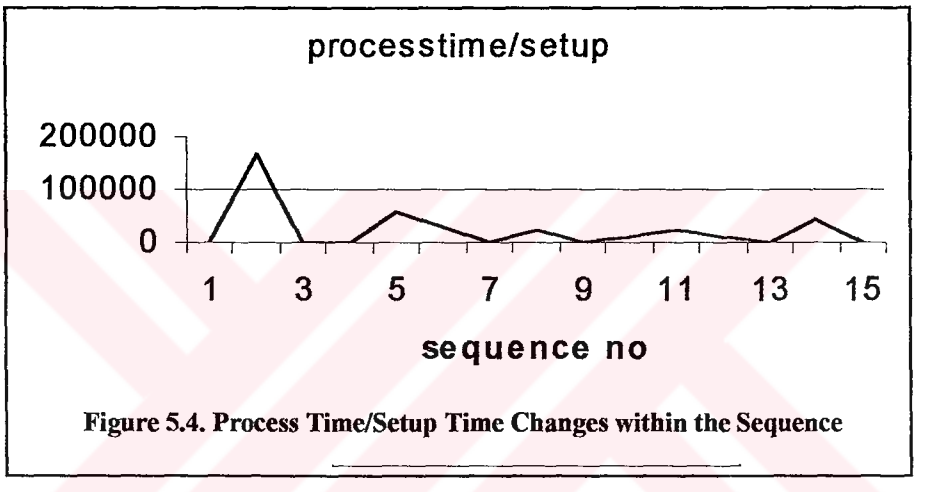


Figure 5.4. Process Time/Setup Time Changes within the Sequence

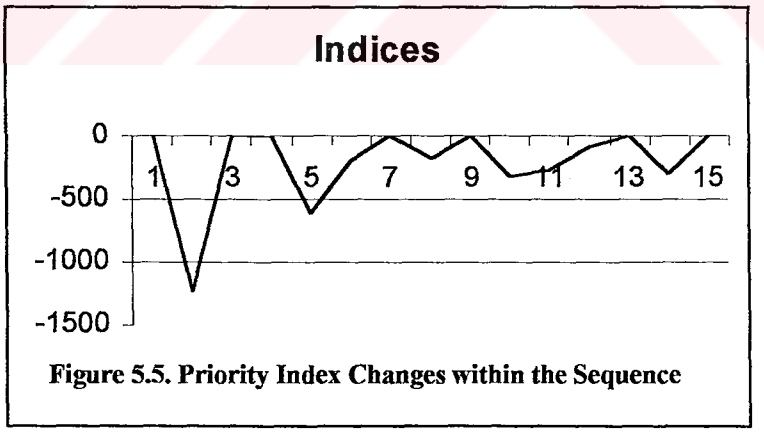


Figure 5.5. Priority Index Changes within the Sequence

As seen in Figures 5.3 through 5.5, the proposed algorithm selects best group in each phase and indices increase after the production of groups, which is the best groups in the previous phase.

CHAPTER SIX
CONCLUSION

Short-term plans are important because they indicate how efficient the resources are used. Machine utilization, setup times, and other performance measures related to the manufacturing environment affect the effectiveness of the planning. This study presents a solution method for a specific manufacturing problem for single machine scheduling. The machine under consideration processes a number of products per cycle, thus the problem is to find the product groups and their sequences for a short term of at most one month. The major goals of this implementation are to maximize the machine utilization and minimize the setup times for each cycle. For this purpose, a priority index was developed including three performance measures:

- Total volume of a particular group (chromosome)
- Process time / setup time ratio.
- Coefficient of variation for each group (chromosome)

For most manufacturing environments, mathematical programming models are too complex and time consuming to implement, because even the simplest planning or scheduling problem may include over 100 constraints. As mentioned in chapter 4, heuristic techniques are most common tools for any planning phase. The characteristics of these methods differentiate according to the processes. The common disadvantage of heuristics is that they do not guarantee the optimum solution, but they provide multiple nearest optimum solutions. One of these heuristic techniques is Genetic Algorithms and includes many types. For this work, genetic algorithms are proved to be the most appropriate tool from the viewpoint of problem characteristics.

Product loads were considered as chromosomes of 6 locuses, which are the candidates for the optimal solution in the genetic algorithm. Each locus may include one of the products in the plan or is zero.

A genetic algorithm procedure starts with creating an initial population. In this study, the initial population consisted of all combinations of the product loads, i.e., chromosomes, including at most 6 products that met the volume constraints.

The second step developed the heuristic function called "evaluation function" or "fitness function" based on a priority rule, which was obtained by discriminant analysis of the three measures (volume, process time/setup time, coefficient of variation of a product group). Evaluation function was developed by using the values which were normalized by their average values for the first time. The evaluation function was the objective function to be minimized separately in each phase of the algorithm. A general function was obtained for all products in section 5.3, and this provided time, hardware and software advantages.

The third step, the selection of the chromosomes, was implemented on the evaluation function (priority index) in each phase of the algorithm. The best group, i.e., the minimum function value, was then selected.

In this study, partially-mapped crossover (explained in section 4.1.5 and 5.4) was implemented in each phase of the sequencing process based on the type of loads. Each setup can be considered as a different sequencing phase so that the best product group (or chromosome) can be selected for the optimum sequence. The algorithm started with the load that had the minimum value. When one of the products was finished, it was removed from the machine. It was then decided which group would be the next through the crossover process. Crossover point(s) were determined based on the blank position in the group (indicates locuses in the chromosome in GA literature). The groups exchanged their products based on the available locuses in the group, so new children were then born. Thus, for each phase, the crossover point varies. The new children (loads) were tested whether they met the volume constraint,

and the children who could be a member in the solution were added to the population, while the others were disposed. After this process, the new generation went on the sequencing process.

Mutation operator was also used in the proposed model. If the same children were born in the last generation, mutation was implemented. Duplicated children were then disposed from the generation. The existence of the duplicated children can be tested any time, so there is no need to use any probability function for the mutation process.

The algorithm was terminated when all the products ordered in the planning period were completed. For each generation, priority indices were calculated to select the minimum one.

The algorithm was executed by using Visual Basic modules of MS-Access. Groups (or Chromosomes) were presented as the records of the tables and queries were used as the objects of the program. Fifteen parts were added to the record sets of the computer program. Pseudo code of the Visual Basic program is presented in *Appendix B*. As seen in Figures 5.3, 5.4, and 5.5, the results obtained from the computer program were very significant and contains systematic flow in itself. Since the products processed on the related machine had same characteristics, the evaluation function, called the priority index, can be generalized for all products. In addition, the algorithm repeats optimization procedure with remaining data in each sequencing phase. Therefore, the record sets in the database can be updated, and execute the algorithm at any time of the planning procedure.

This algorithm can be upgraded for multiple machines with the same characteristics. The model can also be used for group technology or related cellular manufacturing problems, by representing the problem as given in the algorithm and changing the measures within the index, if needed.

REFERENCES

1. Almeida, M.T.,Centeno, M. 1998. A Composite Heuristic for The Single Machine Early/Tardy Job Scheduling Problem. Computers and Operations Research. Vol.25, No.7-8,pp. 625-635.

2. Altenberg.L. 1995. The Schema Theorem and Price's Theorem. In Foundations of Genetic Algorithms 3, San Francisco, California, and USA: Morgan Kaufmann Publishers, pp23- 49.

3. Bierwirth, C., Mattfeld, D. 1999. Production Scheduling and Rescheduling with Genetic Algorithms. Evaluationary Computation 7(I), pp 1-17,MIT .

4. Biskup, D.Chang Edwin, T.C, 1999. Multiple Machine Scheduling With Earliness /Tardiness And Completion Time Penalties. Computers and Operations Research, Vol.26, Pp.45-57.

5. Brandimarte, P. (1995). Advance Models for Manufacturing Systems Management. Florida: CRC Press.

6. Chretienne, P., E. Coffman, J.K.Lenstra. (1997). Scheduling Theory and Its Applications. England, John Wiley and Sons.

7. Corne, D, P. Ross and H.L. Fang. 1994 Fast Practical Evolutionary Timetabling. Proceeding of the AISB Workshop on Evolutionary Computation, Springer Verlag.

8. Daniels, L.R., Stella, Y.H., Webster, S., 1999. Heuristics For Parallel Machine Flexible Resources Scheduling Problems With Unexpected Job Assignment. Computers And Operations Research, Vol.26, Pp 143-155.

9. Figielska, Ewa, 1999. Preemptive Scheduling To Minimize With Changeovers Using Column Generation Technique and Genetic Algorithm. Computers And Industrial Engineering, Vol.37, Pp.81-84.

10. Ghedjati, Fatima, 1999. Genetic Algorithms For The Job shop Scheduling Problem With Unrelated Parallel Constraints: Heuristic Mixing Method Machines And Precedence. Computers And Industrial Engineering, Vol.37, Pp.39-42.

11. Ghrayeb, Omar A. 2000. Solving Jobshop Scheduling Problems with Fuzzy Durations Using Genetic Algorithms, PhD. Thesis. New Mexico State University, Las Cruces, New Mexico.

12. Glover, F. et al, 1993. A User Guide to Tabu Search.

13. Goldberg.D.E.1989. Genetic Algorithms in Search, Optimisation and Machine Learning. Addison-Wesley.

14. Gruau. F.1992. Cellular Encoding of Genetic Neural Networks. Technical Report, rapport No 92-21, Ecole Normale Superieure de Lyon.

15. Kim, J, Kim, Y, 1996. Simulated Annealing And Genetic Algorithm For Scheduling Products With Multilevel Product Structure. Computers And Industrial Engineering, Vol.23, Pp.857-868.

16. Kimms, A., 1999. A Genetic Algorithm For Multilevel Multimachine Lot Sizing And Scheduling. Computers And Industrial Engineering, Vol.26, Pp.829-848.

17. Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi. 1983. Optimization by Simulated Annealing. Science, 220(4598): 671-80.

18. Kitano,H.1990. Designing Neural Networks Using Genetic Algorithms with Graph Generation System. Complex Systems, 4, pp461-476.

19. Kyparisis, G. Koulanas, C, 2000. Open Shop Scheduling with Makespan and Total Completion Time Criteria. Computers And Operations Research, Vol.27, Pp 15-27.

20. Lee, I., Show, M.J., 2000. A Neural Net Approach To Real Time Flow Shop Sequencing. Computers And Industrial Engineering, Vol.38, Pp.125-147.

21. Li Eric, Y.C., Shaw, W.H., 1998. Simulation Modeling Of A Dynamic Job Shop Rescheduling with Machine Availability Constraints. Computers And Industrial Engineering, Vol.25, No.1-2, Pp.117-120.

22. Li Yang, P., Chorn, M.C., 1999. A Generalized Two Machine Flow shop Scheduling Problem with Processing Time Linearly Dependent On Job Waiting –Time. Computers And Industrial Engineering, Vol.38, Pp.365-378.

23. Liaw, Ching Feng, 1999. A Branch And Bound Algorithm For The Single Machine Earliness –Tardiness Scheduling Problem. Computers And Operations Research, Vol.26, Pp 679-693.

24. Lim, SC. 1997. A Genetic Algorithm Based Scheduling System for Dynamic Job-Shop Scheduling Problems, PhD.Thesis, Michigan State University.

25. Linn, R., Zhang, W., 1999. Hybrid Flow Shop Scheduling: A Survey. Computers And Industrial Engineering, Vol.37, Pp. 51-61.

26. Liu, J., Tang, L., 1999. A Modified Genetic Algorithm For Single Machine Scheduling. Computers And Industrial Engineering, Vol.36, Pp.43-46.

27. Liu, Jin, 1999.The Impact Of Neighborhood Size On The Process Of SA Computational Experiments On The Flow Shop Scheduling Problem. Computers And Industrial Engineering, Vol.37, Pp.285-288.

28. Mercado, R.Z.R; Bard, J.F. 1998. Computational Experience with a Branch and Cut Algorithm for Flowshop Scheduling with Setups. Computers and Operations Research. Vol.25, No.5, pp 351-366.

29. Miller, B.L., D.E. 1995. Goldberg. Genetic Algorithms, Selection Schemes and the Varying Effect of Noise. IlliGAL report No. 95009. 1995.

30. Mitchell, M.1996. An Introduction to Genetic Algorithms. London: A Bradford Book, The MIT Press.

31. Morton, Thomas, David W.P.1993. Heuristic Scheduling Systems, Wiley and Sons.

32. Nahmias, S. 2001. Production and Operations Analysis. Singapore, McGraw-Hill, 4TH edition.

33. Nadeen, A.C.2000. A Model for Production Scheduling and Sequencing, PhD.Thesis, University of Northern. Iowa.

34. Nolfi, J.L. Elman and D.Parisi.1994. Learning and Evolution in Neural Networks. Institute of Psychology, C.N.R.-Rome. Technical Report 94-08.

35. O'Reilly, U. 1995. An Analysis of Genetic Programming. PhD Thesis, Carleton University, School of Computer Science.

36. Parunak. 1990. Characterizing Manufacturing Scheduling Problem.

37. Park, Y., Kim, S., Lee, Y., 2000. Scheduling Jobs On Parallel Machines Applying Neural Network and Heuristic Rules. Computers And Industrial Engineering, Vol.38, Pp.189-202.

38. Pinedo, M. (1995). Scheduling, Theory, Algorithms, and Systems. New Jersey, Prentice- Hall.

39. Reeves, Colin. 1995. Modern Heuristic Techniques for Combinatorial Problems, Mphil, McGraw-Hill, London.

40. Reeves, Colin. 1997. Genetic Algorithms for Operations Researcher, INFORMS Journal on Computing; vol9/03,p 242.

41. Ross, P, D. Crone, H.L. Fang. 1994. Successful Lecture Time-tabling with Evolutionary Algorithms. Technical Report AIGA-005-94 (submitted to ECAI workshop on Applications of Evolutionary Algorithm), Department of Artificial Intelligence, University of Edinburgh.

42. Sakawa, M. Mori, T., 1999. An Efficient Genetic Algorithm For Job Shop Scheduling Problems With Fuzzy Processing Time And Fuzzy Due Date. Computers And Industrial Engineering, Vol.36, Pp.325-341.

43. Song, J-S, Lee,T-E. 1998. Petri net Modeling and Scheduling for Cyclic Job Shops with Blocking. Computers and Industrial Engineering. Vol.34, No:2, pp. 281-295.

44. Teressa, A, Conteno, M., 1998. A Composite Heuristic For The Single Machine Early –Tardy Job Scheduling Problem. Computers And Operations Research, Vol.25, pp 625-635.

45. Vollmann, Berry, Whyberk.1997. Manufacturing Planning and Control Systems, McGrawwHill,4th Edition.

46. Wall, M.B. (1996). A Genetic Algorithm for Resource –Constraint Scheduling, PhD Thesis, Massachusetts Institute of Technology.

47. Whitley , D. (1993) A Genetic Algorithm Tutorial. Technical Report CS-93-103, Colorado State University.

48. Wolpert, D.H., W.G. Macready. 1995. No free lunch Theorems for Search. Technical Report SFI-TR-95-02-010, Santa Fe Institute.

49. www.epcc.ac.uk

BIBLIOGRAPHY

1. Alain, Guinet, 2000.Efficiency Of Reductions Of Job shop To Flow shop Problems. European Journal Of Operational Research, No125, Pp. 469-485.

2. Aneja, Y.P., H.Kamoun. 1999. Scheduling of Parts and Robot Activities in a Two Machine Robotic Cell. Computers and Operations Research. Vol.26, pp.297-312.

3. Armentano, V., Ronconi, D.P., 1999. Tabu Search For Total Tardiness Minimization in Flow shop Scheduling Problem. Computers And Operations Research, Vol.26, Pp. 219-235.

4. Bahroun, Z., Baptiste, P., Moalla, M. Lampagne, J.P., 1999. Production Planning And Scheduling in The Context Of Cyclic Delivery Schedulers. Computers And Industrial Engineering, Vol.37, Pp. 3-7.

5. Celena G., S.Fischera, V.Grasso, V.La.Camera, G.Perrane, 1999.An Evolunationary Approach To Multiobjective Scheduling Of Mixed Model Assembly Lines, Computers And Industrial Engineering, Vol.37, Pp.69-73.

6. Chan, F.T.S., Chan, H.K., Lau, H.C.W., 2002. The State Of Art in Simulation Study On FMS Scheduling. A Comprehensive Survey. The International Journal Of Manufacturing Technology, Vol.19, Pp.830-849.

7. Chase, R.B., N.J. Aquilano, R.F.Jacobs. (1998). Operations Management for Competitive Advantage. International Edition, McGraw-Hill.

8. Eric,Y-C. W. Shaw. 1998. Simulation Modeling of a Dynamic Job Shop Rescheduling with Machine Availability Constraints. Computers and Industrial Engineering. Vol:35, No:1-2, pp. 117-120.

9. Ferrel, W, Sale, J., Sams, J., Yellamraju, M., 2000. Evaluating Simple Scheduling Rules in A Mixed Shop Environment. Computers And Industrial Engineering, Vol.38, Pp.39-66.

10. Fığlalı, A., O. Engin (2002). Akış Tipi Çizelgeleme Problemlerinin Genetik Algoritma ile Çözüm Performansının Arttırılmasında Deney Tasarımı Uygulaması. Endüstri Mühendisliği, vol 13, No:3, pp.2-7.

11. Holthaus, Oliver. 1999. Scheduling in Job Shops with Machine Breakdowns : An Experimental Study. Computers and Industrial Engineering. Vol.36, pp. 137-162.

12. Kogan, K, Louner, E., 1998. A Polynomial Algorithm For Scheduling Small Scale Manufacturing Cells Served By Multiple Robots. Computers And Operations Research, Vol.25, No.1, Pp 53-62.

13. Kogen, K., 2000.Scheduling Concurrent Production Over A Finite Planning Horizon: Polinomially Solvable Cases. Computers And Operations Research, Vol.27, Pp 1409-1419.

14. Lee, Fuh-Der Chou Ching-En, 1999. Two Machine Flow Shop Scheduling with Bicriteria Problem Computers And Industrial Engineering, Vol.30, and Pp.549-564.

15. Lee, Sans, J., T, 1998. Petrinet Modeling and Scheduling For Cyclic Job Shop with Blocking Computers And Industrial Engineering, Vol.34, Pp.281-295.

16. Mohamed, N.S. 1998. Operations Planning and Scheduling Problems in an FMS: An Integrated Approach. Computers and Industrial Engineering. Vol:35, No:3-4, pp 443-446.

17. Murata T, H.Ishibutchi, H.Tonaka, 1996. Multi-Objective Genetic Algorithm And Its Applications To Flow Shop Problems. Computers And Industrial Engineering, Vol.30, No.4, Pp.957-968.

18. Parthasarathy, S.; Rajendran, C., 1998. Scheduling To Minimize Mean Tardiness And Weighted Mean Tardiness in Flow shop And Flow Line Based Manufacturing Cell. Computers And Industrial Engineering, Vol.34, and Pp.531-546.

19. Pesch, E., Tetzlaff, U. 1996. Contraint Propagation Based Scheduling of Job Shops. INFORMS Journal on Computing.

20. Rajendran, C., Ziegler, H., 1999. Heuristics For Scheduling in Flow Shops and Flow Line Based Manufacturing Cells To Minimize The Sum of Weighted Flow Time and Weighted Tardiness of Jobs. Computers And Industrial Engineering, Vol.37, Pp.671-690.

21. Render, B. (2003). Quantitative Analysis for Management. International Edition. Prentice-Hall.

22. Santos, D.L.Health, A., 1997. An Application Of Multi Processor Scheduling Methods in The Production of Digital Holographic Images. Computers And Industrial Engineering, Vol.33, No.1-2, Pp.285-288.

23. Schumacher, J., Verwater, Z.Lukazo, Weigen, Mpc., 1999. Disturbances and Their Impact on Scheduling. Computers And Industrial Engineering, Vol.37, Pp.75-79.

24. Schulz, Heady, R.B., 2000. Minimizing The Sum Of Earliness/Tardiness in Multi-machine Scheduling: Mixed Integer Programming Approach. Computers And Industrial Engineering, Vol.38, Pp.297-307.

25. Silver, E.A., D. Pyke, R. Peterson. (1998). Inventory Management, Production Planning and Scheduling. US, John Wiley and Sons.

26. Todd, D., Sen, P., 1999. Distributed Task Scheduling, And Allocation Using GAs. Computers And Industrial Engineering, Vol.37, Pp.47-50.

27. Wall, M.B. (1996). A C++ Library of Genetic Algorithm Components, Massachusetts Institute of Technology.

28. Werra,D, A.Hertz. 1989. Tabu Search Tecniques: A Tutorial to Neural Networks. OR Spectrum. No: 11, pp 131-141.

29. Ulusoy,G, Sivrikaya, F., Bilge, Ü. 1996. A Genetic Algorithm Approach to the Simultaneous Scheduling of Machines and Automatic Guided Vehicles. Computers and Operations Research . Vol.24, No.4, pp. 335-351.

30. www.garage.org

APPENDICES

APPENDIX A: Sample of 6576 Members of Population

ID	PR1	PR2	PR3	PR4	PR5	PR6	VOL	PTIME	SETUP
54257	0	0	0	0	0	P9	2000	4950	35
54204	0	0	0	0	P7	P7	2000	8800	66
54163	0	0	0	0	P3	P8	1750	6600	55
54175	0	0	0	0	P4	P8	2000	6600	60
53731	0	0	0	P3	P8	P10	1900	6600	81
54185	0	0	0	0	P5	P7	1750	17600	63
54256	0	0	0	0	0	P8	1500	6600	32
53777	0	0	0	P4	P5	P5	2000	5500	88
51984	0	0	P3	P8	P10	P15	1990	6600	104
53736	0	0	0	P3	P8	P15	1840	6600	78
51983	0	0	P3	P8	P10	P14	1960	6600	104
54215	0	0	0	0	P8	P10	1650	6600	58
53699	0	0	0	P3	P5	P5	1750	5500	83
54184	0	0	0	0	P5	P6	1450	11000	60
53735	0	0	0	P3	P8	P14	1810	6600	78
54029	0	0	0	P8	P10	P15	1740	6600	81
51998	0	0	P3	P8	P14	P15	1900	6600	101
53778	0	0	0	P4	P5	P6	1950	11000	88
53700	0	0	0	P3	P5	P6	1700	11000	83
51803	0	0	P3	P5	P5	P10	1900	5500	109
53788	0	0	0	P4	P6	P6	1900	4126	88
51236	0	0	P2	P3	P8	P15	2000	6600	101
53701	0	0	0	P3	P5	P7	2000	17600	86
45070	0	P3	P5	P5	P10	P15	1990	5500	132
52237	0	0	P4	P6	P6	P15	1990	4400	111
51808	0	0	P3	P5	P5	P15	1840	5500	106
54183	0	0	0	0	P5	P5	1500	5500	60
54220	0	0	0	0	P8	P15	1590	6600	55
51813	0	0	P3	P5	P6	P10	1850	11000	109
53588	0	0	0	P2	P3	P8	1910	6600	78
53915	0	0	0	P6	P6	P12	1940	4126	88
45125	0	P3	P5	P6	P10	P15	1940	11000	132
53916	0	0	0	P6	P6	P13	1980	4126	88
53202	0	0	P8	P10	P14	P15	1800	6600	104
53848	0	0	0	P5	P5	P10	1650	5500	86
51235	0	0	P2	P3	P8	P14	1970	6600	101
45069	0	P3	P5	P5	P10	P14	1960	5500	132
54028	0	0	0	P8	P10	P14	1710	6600	81
53710	0	0	0	P3	P6	P6	1650	5500	83
52236	0	0	P4	P6	P6	P14	1960	4126	111
51868	0	0	P3	P6	P6	P10	1800	5500	109
51529	0	0	P2	P8	P10	P15	1900	6600	104

51807	0	0	P3	P5	P5	P14	1810	5500	106
42560	0	P2	P3	P5	P5	P15	2000	5500	129
51818	0	0	P3	P5	P6	P15	1790	11000	106
51199	0	0	P2	P3	P5	P5	1910	5500	106
52781	0	0	P6	P6	P12	P14	2000	4126	111
45345	0	P3	P6	P6	P10	P15	1890	5500	132
53853	0	0	0	P5	P5	P15	1590	5500	83
51749	0	0	P3	P4	P6	P12	1990	8250	109
45084	0	P3	P5	P5	P14	P15	1900	5500	129
43954	0	P2	P8	P10	P14	P15	1960	6600	127
42620	0	P2	P3	P6	P6	P10	1960	5500	132
51873	0	0	P3	P6	P6	P15	1740	5500	106
52498	0	0	P5	P5	P10	P15	1740	5500	109
23851	P3	P5	P6	P10	P14	P15	2000	11000	155
54043	0	0	0	P8	P14	P15	1650	6600	78
45124	0	P3	P5	P6	P10	P14	1910	11000	132
53640	0	0	0	P2	P8	P10	1810	6600	81
45344	0	P3	P6	P6	P10	P14	1860	5500	132
42559	0	P2	P3	P5	P5	P14	1970	5500	129
42625	0	P2	P3	P6	P6	P15	1900	5500	129
51210	0	0	P2	P3	P6	P6	1810	5500	106
51528	0	0	P2	P8	P10	P14	1870	6600	104
24566	P3	P6	P6	P10	P14	P15	1950	5500	155
51654	0	0	P3	P3	P5	P5	2000	5500	106
46849	0	P4	P6	P10	P12	P15	1980	8250	135
53858	0	0	0	P5	P6	P10	1600	11000	86
51872	0	0	P3	P6	P6	P14	1710	5500	106
54195	0	0	0	0	P6	P7	1700	17600	63
43250	0	P2	P5	P5	P10	P15	1900	5500	132
45359	0	P3	P6	P6	P14	P15	1800	5500	129
53867	0	0	0	P5	P7	P10	1900	17600	89
53711	0	0	0	P3	P6	P7	1950	17600	86
52497	0	0	P5	P5	P10	P14	1710	5500	109
42624	0	P2	P3	P6	P6	P14	1870	5500	129
54255	0	0	0	0	0	P7	1000	17600	33
51348	0	0	P2	P5	P5	P10	1810	5500	109
15167	P2	P3	P6	P6	P14	P15	1960	5500	152
53860	0	0	0	P5	P6	P12	1990	11000	88
51665	0	0	P3	P3	P6	P6	1900	4126	106
53852	0	0	0	P5	P5	P14	1560	5500	83
51543	0	0	P2	P8	P14	P15	1810	6600	101
47635	0	P5	P5	P10	P14	P15	1800	5500	132
45139	0	P3	P5	P6	P14	P15	1850	11000	129
53645	0	0	0	P2	P8	P15	1750	6600	78
51817	0	0	P3	P5	P6	P14	1760	11000	106
51747	0	0	P3	P4	P6	P10	1600	8250	107

53913	0	0	0	P6	P6	P10	1550	4676	86
54194	0	0	0	0	P6	P6	1400	4126	60
44839	0	P3	P4	P6	P10	P15	1690	8250	130
52778	0	0	P6	P6	P11	P15	1990	4400	111
44445	0	P3	P3	P6	P6	P15	1990	4400	129
42570	0	P2	P3	P5	P6	P15	1950	11000	129
45006	0	P3	P4	P12	P13	P15	1960	5500	130
52512	0	0	P5	P5	P14	P15	1650	5500	106
17443	P2	P5	P5	P10	P14	P15	1960	5500	155
51353	0	0	P2	P5	P5	P15	1750	5500	106
43249	0	P2	P5	P5	P10	P14	1870	5500	132
14647	P2	P3	P4	P6	P10	P15	1850	8250	153
53863	0	0	0	P5	P6	P15	1540	11000	83
52264	0	0	P4	P6	P10	P12	1890	8250	112
46848	0	P4	P6	P10	P12	P14	1950	8250	135
51200	0	0	P2	P3	P5	P6	1860	11000	106
52553	0	0	P5	P6	P10	P15	1690	11000	109
53849	0	0	0	P5	P5	P11	2000	5500	88
54219	0	0	0	0	P8	P14	1560	6600	55
45005	0	P3	P4	P12	P13	P14	1930	5500	130
53679	0	0	0	P3	P3	P8	2000	6600	78
52777	0	0	P6	P6	P11	P14	1960	4126	111
46851	0	P4	P6	P10	P13	P14	1990	8250	135
22850	P3	P4	P6	P10	P14	P15	1750	8250	153
53872	0	0	0	P5	P7	P15	1840	17600	86
52265	0	0	P4	P6	P10	P13	1930	8250	112
44444	0	P3	P3	P6	P6	P14	1960	4126	129
53608	0	0	0	P2	P5	P5	1660	5500	83
51789	0	0	P3	P4	P12	P13	1870	5500	107
52773	0	0	P6	P6	P10	P15	1640	4676	109
51752	0	0	P3	P4	P6	P15	1540	8250	104
53918	0	0	0	P6	P6	P15	1490	4400	83
53914	0	0	0	P6	P6	P11	1900	4126	88
43264	0	P2	P5	P5	P14	P15	1810	5500	129
51413	0	0	P2	P6	P6	P10	1710	4676	109
53689	0	0	0	P3	P4	P6	1450	8250	81
44838	0	P3	P4	P6	P10	P14	1660	8250	130
42499	0	P2	P3	P4	P6	P10	1760	8250	130
20835	P3	P3	P4	P6	P10	P15	1940	8250	153
46879	0	P4	P6	P12	P14	P15	1890	8250	132
52276	0	0	P4	P6	P12	P15	1830	8250	109
51352	0	0	P2	P5	P5	P14	1720	5500	106
52598	0	0	P5	P7	P10	P15	1990	17600	112
29374	P4	P6	P10	P11	P14	P15	2000	8250	158
45484	0	P3	P6	P10	P12	P15	1730	8250	130
45340	0	P3	P6	P6	P10	P10	1950	5500	135

14646	P2	P3	P4	P6	P10	P14	1820	8250	153
52772	0	0	P6	P6	P10	P14	1610	4676	109
51748	0	0	P3	P4	P6	P11	1950	8250	109
15295	P2	P3	P6	P10	P13	P15	1930	8250	153
46885	0	P4	P6	P13	P14	P15	1930	8250	132
45487	0	P3	P6	P10	P13	P15	1770	8250	130
43525	0	P2	P6	P6	P10	P15	1800	4676	132
15292	P2	P3	P6	P10	P12	P15	1890	8250	153
43028	0	P2	P4	P6	P12	P15	1990	8250	132
42569	0	P2	P3	P5	P6	P14	1920	11000	129
51334	0	0	P2	P4	P12	P13	1780	4126	107
43156	0	P2	P4	P10	P12	P13	1930	4676	133
25022	P3	P6	P10	P13	P14	P15	1830	8250	153
52279	0	0	P4	P6	P13	P15	1870	8250	109
25016	P3	P6	P10	P12	P14	P15	1790	8250	153
53644	0	0	0	P2	P8	P14	1720	6600	78
46845	0	P4	P6	P10	P11	P15	1940	8250	135
17109	P2	P4	P10	P12	P13	P14	1990	4676	156
23284	P3	P4	P10	P11	P12	P14	2000	5500	156
44971	0	P3	P4	P10	P11	P12	1940	5500	133
44972	0	P3	P4	P10	P11	P13	1980	5500	133
44784	0	P3	P4	P5	P10	P15	1740	11000	130
53619	0	0	0	P2	P6	P6	1560	4126	83
43524	0	P2	P6	P6	P10	P14	1770	4676	132
44853	0	P3	P4	P6	P14	P15	1600	8250	127
52404	0	0	P4	P10	P12	P13	1770	4676	110
51418	0	0	P2	P6	P6	P15	1650	4400	106
42504	0	P2	P3	P4	P6	P15	1700	8250	127
53917	0	0	0	P6	P6	P14	1460	4126	83
23341	P3	P4	P11	P13	P14	P15	1980	5500	153
24567	P3	P6	P6	P10	P15	P15	1980	5500	155
51900	0	0	P3	P6	P10	P12	1640	8250	107
47855	0	P5	P6	P10	P14	P15	1750	11000	132
47302	0	P4	P10	P12	P13	P15	1860	4676	133
48570	0	P6	P6	P10	P14	P15	1700	4676	132
14661	P2	P3	P4	P6	P14	P15	1760	8250	150
49820	0	P8	P10	P10	P14	P15	1950	6600	130
45809	0	P3	P8	P14	P15	P15	1990	6600	124
43305	0	P2	P5	P6	P10	P15	1850	11000	132
51901	0	0	P3	P6	P10	P13	1680	8250	107
23335	P3	P4	P11	P12	P14	P15	1940	5500	153
53834	0	0	0	P4	P12	P13	1620	4126	84
47301	0	P4	P10	P12	P13	P14	1830	4676	133
44993	0	P3	P4	P11	P12	P15	1880	5500	130
18378	P2	P6	P6	P10	P14	P15	1860	4676	155
43186	0	P2	P4	P12	P13	P15	1870	4400	130

14592	P2	P3	P4	P5	P10	P15	1900	11000	153
44996	0	P3	P4	P11	P13	P15	1920	5500	130
53922	0	0	0	P6	P7	P10	1850	17600	89
14420	P2	P3	P3	P10	P12	P13	1930	4676	151
51751	0	0	P3	P4	P6	P14	1510	8250	104
20834	P3	P3	P4	P6	P10	P14	1910	8250	153
42536	0	P2	P3	P4	P11	P12	1950	5500	130
30606	P4	P10	P12	P13	P14	P15	1920	4676	156
52787	0	0	P6	P6	P14	P15	1550	4400	106
42537	0	P2	P3	P4	P11	P13	1990	5500	130
15168	P2	P3	P6	P6	P15	P15	1990	5500	152
22630	P3	P4	P5	P10	P14	P15	1800	11000	153
21480	P3	P3	P6	P10	P12	P15	1980	8250	153
52552	0	0	P5	P6	P10	P14	1660	11000	109
43185	0	P2	P4	P12	P13	P14	1840	4126	130
15291	P2	P3	P6	P10	P12	P14	1860	8250	153
15294	P2	P3	P6	P10	P13	P14	1900	8250	153
52434	0	0	P4	P12	P13	P15	1710	4400	107
52433	0	0	P4	P12	P13	P14	1680	4126	107
52267	0	0	P4	P6	P10	P15	1440	8250	107
51417	0	0	P2	P6	P6	P14	1620	4126	106
53871	0	0	0	P5	P7	P14	1810	17600	86
47272	0	P4	P10	P10	P12	P13	1920	4126	136
44995	0	P3	P4	P11	P13	P14	1890	5500	130
45483	0	P3	P6	P10	P12	P14	1700	8250	130
51737	0	0	P3	P4	P5	P10	1650	11000	107
43027	0	P2	P4	P6	P12	P14	1960	8250	132
44992	0	P3	P4	P11	P12	P14	1850	5500	130
45486	0	P3	P6	P10	P13	P14	1740	8250	130
43030	0	P2	P4	P6	P13	P14	2000	8250	132
17663	P2	P5	P6	P10	P14	P15	1910	11000	155
54253	0	0	0	0	0	P5	750	11000	30
17173	P2	P4	P12	P13	P14	P15	1930	4400	153
53859	0	0	0	P5	P6	P11	1950	11000	88
52597	0	0	P5	P7	P10	P14	1960	17600	112
51189	0	0	P2	P3	P4	P6	1610	8250	104
42652	0	P2	P3	P6	P10	P12	1800	8250	130
52910	0	0	P6	P10	P12	P13	1970	8250	112
45972	0	P3	P11	P12	P13	P15	1960	5500	130
44102	0	P2	P10	P11	P12	P13	1930	4676	133
51912	0	0	P3	P6	P12	P15	1580	8250	104
20021	P2	P10	P11	P12	P13	P14	1990	4676	156
23666	P3	P5	P5	P14	P15	P15	1990	5500	152
14132	P2	P3	P3	P4	P6	P15	1950	8250	150
49081	0	P6	P12	P13	P14	P15	1970	8250	132
53862	0	0	0	P5	P6	P14	1510	11000	83

51358	0	0	P2	P5	P6	P10	1760	11000	109
47621	0	P5	P5	P10	P10	P15	1890	5500	135
52275	0	0	P4	P6	P12	P14	1800	8250	109
24601	P3	P6	P6	P14	P15	P15	1890	5500	152
42653	0	P2	P3	P6	P10	P13	1840	8250	130
44319	0	P3	P3	P4	P6	P10	1850	8250	130
46844	0	P4	P6	P10	P11	P14	1910	8250	135
43539	0	P2	P6	P6	P14	P15	1710	4400	129
31647	P5	P5	P10	P10	P14	P15	1950	5500	158
45085	0	P3	P5	P5	P15	P15	1930	5500	129
44612	0	P3	P3	P10	P12	P13	1770	4676	128
14160	P2	P3	P3	P4	P10	P13	1890	4676	151
20849	P3	P3	P4	P6	P14	P15	1850	8250	150
45971	0	P3	P11	P12	P13	P14	1930	5500	130
53610	0	0	0	P2	P5	P7	1910	17600	86
47365	0	P4	P12	P13	P14	P15	1770	4400	130
45360	0	P3	P6	P6	P15	P15	1830	5500	129
14450	P2	P3	P3	P12	P13	P15	1870	4400	148
51915	0	0	P3	P6	P13	P15	1620	8250	104
52278	0	0	P4	P6	P13	P14	1840	8250	109
52818	0	0	P6	P7	P10	P15	1940	17600	112
51999	0	0	P3	P8	P15	P15	1930	6600	101
53188	0	0	P8	P10	P10	P15	1890	6600	107
49005	0	P6	P10	P11	P12	P15	1980	8250	135
14159	P2	P3	P3	P4	P10	P12	1850	4676	151
15322	P2	P3	P6	P12	P14	P15	1800	8250	150
15328	P2	P3	P6	P13	P14	P15	1840	8250	150
42503	0	P2	P3	P4	P6	P14	1670	8250	127
46628	0	P4	P5	P10	P12	P14	2000	11000	135
40154	0	P1	P3	P4	P10	P15	1990	5500	133
44783	0	P3	P4	P5	P10	P14	1710	11000	130
52263	0	0	P4	P6	P10	P11	1850	8250	112
43955	0	P2	P8	P10	P15	P15	1990	6600	127
49855	0	P8	P10	P14	P15	P15	1890	6600	127
22338	P3	P4	P4	P10	P12	P14	2000	5500	156
44152	0	P2	P11	P12	P13	P15	1870	4400	130
42403	0	P2	P3	P3	P4	P13	1740	4126	125
21482	P3	P3	P6	P10	P13	P14	1990	8250	153
42489	0	P2	P3	P4	P5	P10	1810	11000	130
45514	0	P3	P6	P12	P14	P15	1640	8250	127
45520	0	P3	P6	P13	P14	P15	1680	8250	127
15075	P2	P3	P5	P10	P13	P15	1980	11000	153
21932	P3	P3	P10	P12	P13	P14	1830	4676	151
44981	0	P3	P4	P10	P13	P15	1570	5500	128
21479	P3	P3	P6	P10	P12	P14	1950	8250	153
51294	0	0	P2	P4	P6	P12	1900	8250	109

53927	0	0	0	P6	P7	P15	1790	17600	86
17099	P2	P4	P10	P11	P13	P14	1950	4676	156
17100	P2	P4	P10	P11	P13	P15	1980	4676	156
15072	P2	P3	P5	P10	P12	P15	1940	11000	153
45264	0	P3	P5	P10	P12	P15	1780	11000	130
44352	0	P3	P3	P4	P10	P13	1730	4676	128
15288	P2	P3	P6	P10	P11	P15	1850	8250	153
51784	0	0	P3	P4	P11	P12	1790	5500	107
44978	0	P3	P4	P10	P12	P15	1530	5500	128
53792	0	0	0	P4	P6	P10	1350	8250	84
14591	P2	P3	P4	P5	P10	P14	1870	11000	153
45267	0	P3	P5	P10	P13	P15	1820	11000	130
24307	P3	P5	P10	P13	P14	P15	1880	11000	153
52567	0	0	P5	P6	P14	P15	1600	11000	106
43245	0	P2	P5	P5	P10	P10	1960	5500	135
43152	0	P2	P4	P10	P11	P13	1890	4676	133
17174	P2	P4	P12	P13	P15	P15	1960	4126	153
17444	P2	P5	P5	P10	P15	P15	1990	5500	155
51905	0	0	P3	P6	P11	P12	1990	8250	109
25006	P3	P6	P10	P11	P14	P15	1750	8250	153
45120	0	P3	P5	P6	P10	P10	2000	11000	135
42664	0	P2	P3	P6	P12	P15	1740	8250	127
22375	P3	P4	P4	P13	P14	P15	1980	5500	153
46854	0	P4	P6	P10	P14	P15	1500	8250	130
44351	0	P3	P3	P4	P10	P12	1690	4676	128
17097	P2	P4	P10	P11	P12	P15	1940	4676	156
14789	P2	P3	P4	P10	P13	P15	1730	5500	151
54254	0	0	0	0	0	P6	700	8250	30
44642	0	P3	P3	P12	P13	P15	1710	4400	125
24301	P3	P5	P10	P12	P14	P15	1840	11000	153
51779	0	0	P3	P4	P10	P12	1440	5500	105
51372	0	0	P2	P5	P7	P15	2000	17600	109
17096	P2	P4	P10	P11	P12	P14	1910	4676	156
43304	0	P2	P5	P6	P10	P14	1820	11000	132
21933	P3	P3	P10	P12	P13	P15	1860	4676	151
16662	P2	P4	P6	P10	P14	P15	1660	8250	153
51295	0	0	P2	P4	P6	P13	1940	8250	109
30542	P4	P10	P10	P12	P13	P14	1980	4126	159
44324	0	P3	P3	P4	P6	P15	1790	8250	127
51785	0	0	P3	P4	P11	P13	1830	5500	107
45480	0	P3	P6	P10	P11	P15	1690	8250	130
42667	0	P2	P3	P6	P13	P15	1780	8250	127
22369	P3	P4	P4	P12	P14	P15	1940	5500	153
51651	0	0	P3	P3	P4	P13	1580	4126	102
42402	0	P2	P3	P3	P4	P12	1700	4126	125
43151	0	P2	P4	P10	P11	P12	1850	4676	133

51742	0	0	P3	P4	P5	P15	1590	11000	104
47620	0	P5	P5	P10	P10	P14	1860	5500	135
52940	0	0	P6	P12	P13	P15	1910	8250	109
51330	0	0	P2	P4	P11	P13	1740	4126	107
14786	P2	P3	P4	P10	P12	P15	1690	5500	151
52209	0	0	P4	P5	P10	P12	1940	11000	112
20120	P2	P11	P12	P13	P14	P15	1930	4400	153
46869	0	P4	P6	P11	P14	P15	1850	8250	132
51780	0	0	P3	P4	P10	P13	1480	5500	105
18364	P2	P6	P6	P10	P10	P15	1950	4400	158
43019	0	P2	P4	P6	P10	P15	1600	8250	130
20977	P3	P3	P4	P10	P13	P15	1820	4676	151
43520	0	P2	P6	P6	P10	P10	1860	4126	135
43024	0	P2	P4	P6	P11	P15	1950	8250	132
52612	0	0	P5	P7	P14	P15	1900	17600	109
53794	0	0	0	P4	P6	P12	1740	8250	86
51329	0	0	P2	P4	P11	P12	1700	4126	107
51655	0	0	P3	P3	P5	P6	1950	11000	106
52210	0	0	P4	P5	P10	P13	1980	11000	112
51650	0	0	P3	P3	P4	P12	1540	4126	102
20780	P3	P3	P4	P5	P10	P15	1990	11000	153
20976	P3	P3	P4	P10	P13	P14	1790	4676	151
42531	0	P2	P3	P4	P10	P12	1600	5500	128
14131	P2	P3	P3	P4	P6	P14	1920	8250	150
51738	0	0	P3	P4	P5	P11	2000	11000	109
14416	P2	P3	P3	P10	P11	P13	1890	4676	151
14292	P2	P3	P3	P6	P12	P15	1990	8250	150
17075	P2	P4	P10	P10	P11	P12	2000	4126	159
42532	0	P2	P3	P4	P10	P13	1640	5500	128
46625	0	P4	P5	P10	P11	P15	1990	11000	135
14174	P2	P3	P3	P4	P13	P15	1830	4400	148
31682	P5	P5	P10	P14	P15	P15	1890	5500	155
54161	0	0	0	0	P3	P6	950	8250	53
51363	0	0	P2	P5	P6	P15	1700	11000	106
50213	0	P10	P11	P12	P13	P14	1830	4676	133
53688	0	0	0	P3	P4	P5	1500	11000	81
20974	P3	P3	P4	P10	P12	P15	1780	4676	151
30607	P4	P10	P12	P13	P15	P15	1950	4676	156
27218	P4	P4	P6	P10	P14	P15	2000	8250	158
42481	0	P2	P3	P4	P4	P13	1990	5500	130
16153	P2	P4	P4	P10	P13	P14	1950	4676	156
42480	0	P2	P3	P4	P4	P12	1950	5500	130
51791	0	0	P3	P4	P12	P15	1380	5500	102
45808	0	P3	P8	P14	P14	P15	1960	6600	124
44715	0	P3	P4	P4	P10	P12	1940	5500	133
21516	P3	P3	P6	P13	P14	P15	1930	8250	150

20973	P3	P3	P4	P10	P12	P14	1750	4676	151
53609	0	0	0	P2	P5	P6	1610	11000	83
15745	P2	P3	P10	P12	P13	P15	1770	5500	151
23306	P3	P4	P10	P13	P14	P15	1630	5500	151
47366	0	P4	P12	P13	P15	P15	1800	4126	130
38474	P10	P11	P12	P13	P14	P15	1920	4676	156
21510	P3	P3	P6	P12	P14	P15	1890	8250	150
22836	P3	P4	P6	P10	P10	P15	1840	8250	156
30586	P4	P10	P11	P13	P14	P15	1880	4676	156
44980	0	P3	P4	P10	P13	P14	1540	5500	128
47636	0	P5	P5	P10	P15	P15	1830	5500	132
42896	0	P2	P4	P4	P10	P13	1890	4676	133
44716	0	P3	P4	P4	P10	P13	1980	5500	133
14788	P2	P3	P4	P10	P13	P14	1700	5500	151
47292	0	P4	P10	P11	P13	P15	1820	4676	133
14171	P2	P3	P3	P4	P12	P15	1790	4400	148
14415	P2	P3	P3	P10	P11	P12	1850	4676	151
51794	0	0	P3	P4	P13	P15	1420	5500	102
44798	0	P3	P4	P5	P14	P15	1650	11000	127
40153	0	P1	P3	P4	P10	P14	1960	5500	133
46665	0	P4	P5	P13	P14	P15	1980	11000	132
44977	0	P3	P4	P10	P12	P14	1500	5500	128
52272	0	0	P4	P6	P11	P15	1790	8250	109
54150	0	0	0	0	P2	P8	1660	6600	55
44151	0	P2	P11	P12	P13	P14	1840	3576	130
48556	0	P6	P6	P10	P10	P15	1790	4400	135
50214	0	P10	P11	P12	P13	P15	1860	4676	133
46659	0	P4	P5	P12	P14	P15	1940	11000	132
52266	0	0	P4	P6	P10	P14	1410	8250	107
43175	0	P2	P4	P11	P13	P14	1800	4126	130
47289	0	P4	P10	P11	P12	P15	1780	4676	133
30580	P4	P10	P11	P12	P14	P15	1840	4676	156
53350	0	0	P10	P11	P12	P13	1770	4676	110
18379	P2	P6	P6	P10	P15	P15	1890	4676	155
53187	0	0	P8	P10	P10	P14	1860	6600	107
14785	P2	P3	P4	P10	P12	P14	1660	5500	151
16150	P2	P4	P4	P10	P12	P14	1910	4676	156
42895	0	P2	P4	P4	P10	P12	1850	4676	133
42494	0	P2	P3	P4	P5	P15	1750	11000	127
23300	P3	P4	P10	P12	P14	P15	1590	5500	151
45937	0	P3	P10	P12	P13	P15	1610	5500	128
47268	0	P4	P10	P10	P11	P13	1880	4126	136
44366	0	P3	P3	P4	P13	P15	1670	4400	125
43176	0	P2	P4	P11	P13	P15	1830	4400	130
53714	0	0	0	P3	P6	P10	1100	8250	79
21996	P3	P3	P12	P13	P14	P15	1770	4400	148

42543	0	P2	P3	P4	P12	P15	1540	5500	125
53400	0	0	P11	P12	P13	P15	1710	4400	107
47291	0	P4	P10	P11	P13	P14	1790	4676	133
16154	P2	P4	P4	P10	P13	P15	1980	4676	156
14606	P2	P3	P4	P5	P14	P15	1810	11000	150
53795	0	0	0	P4	P6	P13	1780	8250	86
42546	0	P2	P3	P4	P13	P15	1580	5500	125
52768	0	0	P6	P6	P10	P10	1700	4126	112
20963	P3	P3	P4	P10	P10	P13	1880	4126	154
47288	0	P4	P10	P11	P12	P14	1750	4676	133
14173	P2	P3	P3	P4	P13	P14	1800	4126	148
44299	0	P3	P3	P4	P4	P11	2000	2750	130
51903	0	0	P3	P6	P10	P15	1190	8250	102
44727	0	P3	P4	P4	P12	P15	1880	5500	130
43172	0	P2	P4	P11	P12	P14	1760	4126	130
43173	0	P2	P4	P11	P12	P15	1790	4400	130
42792	0	P2	P3	P10	P12	P13	1680	5500	128
44472	0	P3	P3	P6	P10	P12	1890	8250	130
49004	0	P6	P10	P11	P12	P14	1950	8250	135
21476	P3	P3	P6	P10	P11	P15	1940	8250	153
24565	P3	P6	P6	P10	P14	P14	1920	5500	155
42910	0	P2	P4	P4	P13	P15	1830	4400	130
52399	0	0	P4	P10	P11	P12	1690	4676	110
17153	P2	P4	P11	P13	P14	P15	1890	4400	153
38475	P10	P11	P12	P13	P15	P15	1950	4676	156
43989	0	P2	P8	P14	P15	P15	1900	6600	124
18363	P2	P6	P6	P10	P10	P14	1920	4126	158
52493	0	0	P5	P5	P10	P10	1800	5500	112
44363	0	P3	P3	P4	P12	P15	1630	4400	125
44730	0	P3	P4	P4	P13	P15	1920	5500	130
14822	P2	P3	P4	P13	P14	P15	1640	5500	148
51524	0	0	P2	P8	P10	P10	1960	6600	107
52055	0	0	P3	P11	P12	P13	1870	5500	107
14440	P2	P3	P3	P11	P13	P15	1830	4400	148
49007	0	P6	P10	P11	P13	P14	1990	8250	135
44473	0	P3	P3	P6	P10	P13	1930	8250	130
48735	0	P6	P7	P10	P14	P15	2000	17600	135
16151	P2	P4	P4	P10	P12	P15	1940	4676	156
15744	P2	P3	P10	P12	P13	P14	1740	5500	151
53399	0	0	P11	P12	P13	P14	1680	3576	107
54100	0	0	0	P11	P12	P13	1620	2750	84
26238	P3	P10	P12	P13	P14	P15	1670	5500	151
50312	0	P11	P12	P13	P14	P15	1770	4400	130
52817	0	0	P6	P7	P10	P14	1910	17600	112
53203	0	0	P8	P10	P15	P15	1830	6600	104
44729	0	P3	P4	P4	P13	P14	1890	5500	130

43319	0	P2	P5	P6	P14	P15	1760	11000	129
53797	0	0	0	P4	P6	P15	1290	8250	81
14816	P2	P3	P4	P12	P14	P15	1600	5500	148
47267	0	P4	P10	P10	P11	P12	1840	4126	136
38375	P10	P10	P11	P12	P13	P14	1980	3576	159
44726	0	P3	P4	P4	P12	P14	1850	5500	130
52400	0	0	P4	P10	P11	P13	1730	4676	110
23665	P3	P5	P5	P14	P14	P15	1960	5500	152
34364	P6	P6	P10	P10	P14	P15	1850	4400	158
44365	0	P3	P3	P4	P13	P14	1640	4126	125
45014	0	P3	P4	P13	P14	P15	1480	5500	125
17147	P2	P4	P11	P12	P14	P15	1850	4400	153
23336	P3	P4	P11	P12	P15	P15	1970	5500	153
39570	0	P1	P2	P3	P4	P15	2000	5500	130
51911	0	0	P3	P6	P12	P14	1550	8250	104
15287	P2	P3	P6	P10	P11	P14	1820	8250	153
15074	P2	P3	P5	P10	P13	P14	1950	11000	153
48555	0	P6	P6	P10	P10	P14	1760	4126	135
45008	0	P3	P4	P12	P14	P15	1440	5500	125
14170	P2	P3	P3	P4	P12	P14	1760	4126	148
15071	P2	P3	P5	P10	P12	P14	1910	11000	153
53716	0	0	0	P3	P6	P12	1490	8250	81
53695	0	0	0	P3	P4	P12	1290	5500	79
40168	0	P1	P3	P4	P14	P15	1900	5500	130
20962	P3	P3	P4	P10	P10	P12	1840	4126	154
17478	P2	P5	P5	P14	P15	P15	1900	5500	152
51790	0	0	P3	P4	P12	P14	1350	5500	102
41289	0	P1	P6	P10	P14	P15	2000	8250	135
18828	P2	P6	P10	P12	P14	P15	1700	8250	153
51195	0	0	P2	P3	P4	P12	1450	5500	102
42907	0	P2	P4	P4	P12	P15	1790	4400	130
18834	P2	P6	P10	P13	P14	P15	1740	8250	153
14031	P2	P3	P3	P3	P4	P13	1990	4126	148
52221	0	0	P4	P5	P12	P15	1880	11000	109
15808	P2	P3	P12	P13	P14	P15	1680	5500	148
42822	0	P2	P3	P12	P13	P15	1620	5500	125
14294	P2	P3	P3	P6	P13	P14	2000	8250	150
27641	P4	P4	P10	P10	P13	P15	1970	4400	159
14449	P2	P3	P3	P12	P13	P14	1840	3576	148
16187	P2	P4	P4	P13	P14	P15	1890	4400	153
14291	P2	P3	P3	P6	P12	P14	1960	8250	150
42542	0	P2	P3	P4	P12	P14	1510	5500	125
42545	0	P2	P3	P4	P13	P14	1550	5500	125
14437	P2	P3	P3	P11	P12	P15	1790	4400	148

APPENDIX B : Pseudo Code Of Computer Program For Proposed Method

MODULE_1 (population)

/*This algorithm constitutes all possible chromosomes in a population for those machine including6 fixtures*/

```
BEGIN
Read n
Array (n) 'for n products'
i←1
k←1
l←1
t←1
x←1
y←1
j←1
sira← 0
For j←1 to 6 do
        While not (i=n and k=n and l=n and x=n and y=n+1) do
         If j=1 then
            If y=n+1 and x <n and t<n and l<n and k<n and i<n then
               x←x+1
                              y←x
            Elseif  y=n+1 and x=n and t<n and l<n and k<n and i<n then
               t←t+1
                              x←t
                              y←t
            Elseif y=n+1 and x=n and t=n and l<n and k<n and i<n then
               l←l+1
                              t←l
                              x←l
                              y←l
```

```
Elseif y=n+1 and x=n and t=n and l=n and k<n and i<n then
   k←k+1
   l←k
   t←k
   x←k
   y←k
Elseif y=n+1 and x=n and t=n and l=n and k=n and i<n then
   i←i+1
   k←i
   l←i
   t←i
   x←i
   y←i
Endif
'In that loopthe table identified as PRO is updated by adding the matrix values, where PR1,PR2,PR3,PR4,PR5,PR6,sira constitute each field of PRO table'
  While y <= n  do
PR1←P(i-1)
PR2←P(k-1)
PR3←P(l-1)
PR4←P(t-1)
PR5←P(x-1)
PR6←P(y-1)
sira←sira +1 'value of the sira defined as variable is added to the field called sira in PRO table.
sira←sira +1
if y<=n then y←y+1
End while

If j=2 then
   If y=n+1 and x <n and t<n and l<n and k<n  then
      x←x+1
```

y←x

Elseif y=n+1 and x=n and t<n and l<n and k<n then

t←t+1

x←t

y←t

Elseif y=n+1 and x=n and t=n and l<n and k<n and then

l←l+1

t←l

x←l

y←l

Elseif y=n+1 and x=n and t=n and l=n and k<n then

k←k+1

l←k

t←k

x←k

y←k

i←n

Endif

'In that loopthe table identified as PRO is updated by adding the matrix values, where PR1,PR2,PR3,PR4,PR5,PR6,sira costitute each field of PRO table'

While y <= n do

PR1←0

PR2←P(k-1)

PR3←P(l-1)

PR4←P(t-1)

PR5←P(x-1)

PR6←P(y-1)

sira←sira +1 'value of the sira defined as variable is added to the field called sira in PRO table.

sira←sira+1

if y<=n then y←y+1

End while

If j=3 then

If y=n+1 and x <n and t<n and l<n and then

x←x+1

y←x

Elseif y=n+1 and x=n and t<n and l<n then

t←t+1

x←t

y←t

Elseif y=n+1 and x=n and t=n and l<n then

l←l+1

t←l

x←l

y←l

i←n

k←n

Endif

'In that loopthe table identified as PRO is updated by adding the matrix values, and PR1,PR2,PR3,PR4,PR5,PR6,sira constitute each field of PRO table'

While y <= n do

PR1←0

PR2←0

PR3←P(l-1)

PR4←P(t-1)

PR5←P(x-1)

PR6←P(y-1)

sira←sira +1 'value of the sira defined as variable is added to the field called sira in PRO table.

sira←sira+1

if y<=n then y←y+1

End while

```
If j= 4 then
 If y=n+1 and x <n and t<n then
    x←x+1
                y←x
 Elseif  y=n+1 and x=n and t<n then
    t←t+1
                x←t
                y←t
                l←n
                k←n
                 i←n

 Endif
'In that loopthe table identified as PRO is updated by adding the matrix values, where PR1,PR2,PR3,PR4,PR5,PR6,sira constitute each field of PRO table'
 While y <= n  do
        PR1←0
        PR2←0
        PR3←0
        PR4←P(t-1)
        PR5←P(x-1)
        PR6←P(y-1)
        sira←sira +1  'value of the sira defined as variable is added to the field called sira in PRO table.
        sira←sira+1
        if y<=n then y←y+1
        End while

ElseIf j= 5 then
 If y=n+1 and x <n then
    x←x+1
```

```
                y←x
                t←n
                l←n
                k←n
                i←n
        Endif
```

'In that loopthe table identified as PRO is updated by adding the matrix values, where PR1,PR2,PR3,PR4,PR5,PR6,sira constitute each field of PRO table'

```
        While y <= n  do
                PR1←0
                PR2←0
                PR3←0
                PR4←0
                PR5←P(x-1)
                PR6←P(y-1)
                sira←sira +1  'value of the sira defined as variable is added to the
        field called sira in PRO table.
                sira←sira+1
                if y<=n then y←y+1
                End while

Elseif j=6 Then
        x←n
        t←n
        l←n
                k←n
                i←n
```

'In that loopthe table identified as PRO is updated by adding the matrix values, where PR1,PR2,PR3,PR4,PR5,PR6,sira constitute each field of PRO table'

```
                While y <= n  do
```

PR1←0

PR2←0

PR3←0

PR4←0

PR5←0

PR6←P(y-1)

sira←sira +1 'value of the sira defined as variable is added to the field called sira in PRO table.

sira←sira+1

if y<=n then y←y+1

End while

End if

End while

i←1

k←1

l←1

t←1

x←1

y←1

Next j

END

MODULE_2 (feasable_1)

/*This module sellects the feasable groups (chromosomes) from the population module and calculates heuristic function value of each group*/

/*In feasable module there are 3 tables used as a record set: pro,proset, and products*/

while not pro end of file

'do followingsfrom the first record until the end of the pro table'

setup←0

While not pro!PR1= product!PR do

Move to the next record of product table 'finds the same product in the products table'

End while

volume←product!vp

setup←setup+product!setup

Move to the first record in product table

While not pro!PR2= product!PR do

Move to the next record of product table 'finds the same product in the products table'

End while

volume←volume+product!vp

setup←setup+product!setup

Move to the first record in product table

While not pro!PR3= product!PR do

Move to the next record of product table 'finds the same product in the products table'

End while

volume←volume+product!vp

setup←setup+product!setup

Move to the first record in product table

While not pro!PR4= product!PR do

Move to the next record of product table 'finds the same product in the products table'

End while

volume←volume+product!vp

setup←setup+product!setup

Move to the first record in product table

While not pro!PR5= product!PR do

Move to the next record of product table 'finds the same product in the products table'

End while

volume←volume+product!vp

setup←setup+product!setup

Move to the first record in product table

While not pro!PR6 = product!PR do

Move to the next record of product table 'finds the same product in the products table'

End while

volume←volume+product!vp

setup←setup+product!setup

Move to the first record in product table

If volume <= 2000 Then

ptime←findProcTime(pro!PR1,pro!PR2, pro!PR3, pro!PR4, pro!PR5, pro!PR6)*55

covar←coefvar(pro!PR1,pro!PR2, pro!PR3, pro!PR4, pro!PR5, pro!PR6)

proset!PR1←pro!PR1 'Add the record to proset table'

proset!PR2←pro!PR2

proset!PR3←pro!PR3

proset!PR4←pro!PR4

proset!PR5←pro!PR5

proset!PR6←pro!PR6

proset!vt←volume

proset!sira←pro!sira

```
        proset!setup←setup
        proset!proc_time←ptime
        proset!pro_set←ptime/setup
        proset!coefvar←covar
        proset!indices← -0.73*(ptime/setup)+0.953*covar-0.989*volume
        Update proset table

End if

volume←0
Move the next record of Pro  table.
End while
Sort proset table according to the indice records (ascending)

FUNCTION coefvar(t1,t2,t3,t4,t5,t6)
diffsqr←0
count←0

        If t1 ^= 0 Then
                count←1
                quant_total←findquant(t1)
        End If

        If t2 ^= 0 Then
                count←count + 1
                quant_total ←quant_total + findquant(t2)
        End If

        If t3 ^=0 Then
                count← count + 1
                quant_total ← quant_total + findquant(t3)
        End If
```

```
        If t4 ^=0 Then
                count←count + 1
                quant_total← quant_total + findquant(t4)
        End If

        If t5^=0 Then
                count ←count + 1
                quant_total ← quant_total + findquant(t5)
        End If

        If t6 ^=0 Then
                count← count + 1
                quant_total ←quant_total + findquant(t6)
        End If

ort = quant_total / count

        If t1 ^=0 Then
                diffsqr← (ort - findquant(t1)) ^ 2
        End If

        If t2 ^=0 Then
                diffsqr←(ort - findquant(t2)) ^ 2 + diffsqr
        End If

        If t3 ^=0 Then
                diffsqr← (ort - findquant(t3)) ^ 2 + diffsqr
        End If

        If t4 ^= 0 Then
                diffsqr←(ort - findquant(t4)) ^ 2 + diffsqr
```

```
        End If

        If t5 ^=0 Then
                diffsqr←(ort - findquant(t5)) ^ 2 + diffsqr
        End If

        If t6 ^= "0" Then
                diffsqr = (ort - findquant(t6)) ^ 2 + diffsqr
        End If

        If count > 1 Then
                coefvar← ((diffsqr / (count - 1)) ^ 0.5) / ort
        Else
                coefvar = 0
        End If

End Function
```

FUNCTION findquant(p1)

Find minimum number of products remained (from products table) for current chromosome using a query (cloop) and quantity of related products in the group(cunt field in the query)

```
findquant←remained/cunt
END function
```

FUNCTION findProcTime(u1,u2,u3,u4,u5,u6):

Add each parameter of function as a new record to procount table

Select a query "findloop"

cycle←Find min PR, maxloop from findLoop

END function

MODULE_3 (solution) and MODULE_4 (feasible_2)

/*Solution module starts the solution procedure by choosing the chromosome with minimum indice value. Feasable_2 works in the solution module by using the same codes in the module_2(feasible) to calculate the required measures in each phase of te genetic algorithm.*/

BEGIN

Set proset,products,pro and solution table as recordsets

ptime ← 0

Raw 10:

Delete all data from sol1 table

Sort all data in proset table according to the priority indices (ascending)

Add all data in proset table to sol1 table

If sol =EOF then

Exit solution

End if 'to end the module'

Add the current chromosome(parent_1) and another chromosome (parent_2) to proCount table.

While not sol1 table =EOF

Cross over these two products choosing genes that have finished as crossover points

Calculate cycle count for new children

If the child is a copy of the current chromosome then

Ptime←time + proc_time

Else

ptime←e = ptime+setup + proc_time

End if

End While

End If

Delete the finished parts from all chromosomes as mutation process

Add new children to sol1 table

Add the first group to prod_schedule table

Update products table 'finished orders are subtracted from orders field)

Call feasible2

```
'at this point feasible_2 module works to calculate priority indices for new
population'
GoTo 10
MODULE_4(feasable_2)
While Not pro.EOF
setup ← 0
Move the first record of products table
While No PRO!PR1  =products!PR
Move the next record
volume ← product!vp
        If setup_true(pro!PR1) = False Then
                setup ← setup + product!setupTime
        End If
Move the first record of products table
While No PRO!PR2  =products!PR
Move the next record
volume ← product!vp
        If setup_true(pro!PR2) = False Then
                setup ← setup + product!setupTime
        End If

Move the first record of products table
While No PRO!PR3  =products!PR
Move the next record
volume ← product!vp
        If setup_true(pro!PR3) = False Then
                setup ← setup + product!setupTime
        End If

Move the first record of products table
While No PRO!PR4  =products!PR
Move the next record
```

```
volume ← product!vp
        If setup_true(pro!PR4) = False Then
                setup ← setup + product!setupTime
        End If

Move the first record of products table
While No PRO!PR5  =products!PR
Move the next record
volume ← product!vp
        If setup_true(pro!PR5) = False Then
                setup ← setup + product!setupTime
        End If

Move the first record of products table
While No PRO!PR6  =products!PR
Move the next record
volume ← product!vp
        If setup_true(pro!PR6) = False Then
                setup ← setup + product!setupTime
        End If
Calculate coefvar for each chromosome of new generation  'same with the feasble_1'
Calculate pro_set for each chromosome of new generation 'same with the feasble_1'
Calculate priority indices for each chromosome of new generation       'same with
the feasble_1'

FUNCTION setup_true(x)
   If current chromosome ^= new child
                setup_true ← False
        Else
                setup_true ← True
        End If
End Function
```

APPENDIX C: Discriminant Analysis Syntax of SSPS for Windows

```
DISCRIMINANT GROUPS=varname(min,max)
 /VARIABLES=varlist
 [/SELECT=varname(value)]
 [/ANALYSIS=varlist(level) [varlist...]]
 [/METHOD={DIRECT**}] [/TOLERANCE={0.001}]
   {WILKS  }          {t   }
       {MAHAL  }
       {MAXMINF }
       {MINRESID}
       {RAO    }
 [/MAXSTEPS={2v}]
         {m }
 [/FIN={3.84}] [/FOUT={2.71}] [/VIN={0**}]
     {fi }       {fo }       {vi }
 [/PIN={pi}] [/POUT={po}]
 [/FUNCTIONS={g-1,100.0,1.0**}]
          {nf, cp ,sig }
 [/PRIORS={EQUAL**  }]
       {SIZE    }
       {value list}
 [/SAVE=[CLASS=varname] [PROBS=rootname]
      [SCORES=rootname]]
 [/ANALYSIS=...]
 [/MISSING={EXCLUDE**}]
        {INCLUDE }
 [/MATRIX=[OUT(file)] [IN(file)]]
 [/HISTORY={STEP**}]
        {NONE }
```

[/ROTATE={NONE** }]
{COEFF }

{STRUCTURE}
[/CLASSIFY={NONMISSING }{POOLED }[MEANSUB]]
{UNSELECTED }{SEPARATE}
{UNCLASSIFIED}
[/STATISTICS=[MEAN][COV][FPAIR][RAW]]
[STDDEV][GCOV][UNIVF][COEFF]
[CORR][TCOV][BOXM][TABLE]
[CROSSVALID][ALL]
[/PLOT=[MAP][SEPARATE][COMBINED][CASES[(n)]][ALL]]
[/OUTFILE MODEL(filename)]
**Default if the subcommand is omitted.
Default for /MATRIX OUT and IN is the working data file.