A GENETIC ALGORITHM APPROACH TO THE TOOL ALLOCATION PROBLEM IN FLEXIBLE MANUFACTURING SYSTEMS

by

İlker Güçlü

B.S., Industrial Engineering, Boğaziçi University, 2003

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering
Boğaziçi University

2006

To my family…

ACKNOWLEDGEMENTS

First of all, I am extremely grateful to my thesis supervisor Assoc. Prof. Ümit Bilge for her guidance, support, encouragement and motivation throughout this study.

I would like to express my gratitude to Prof. Tülin Aktin and Assoc. Prof. Necati Aras for taking part in my thesis jury.

I am also grateful to Assoc. Prof. Wolfgang Hörmann for his invaluable comments and suggestions in this study.

I would like to thank my colleague Erinç Albey for his continuous moral support and encouragement.

The work reported in this thesis is supported by Boğaziçi University Research Fund under Grant No. 05A301.

ABSTRACT

A GENETIC ALGORITHM APPROACH TO THE TOOL ALLOCATION PROBLEM IN FLEXIBLE MANUFACTURING SYSTEMS

This thesis focuses on the development of a Genetic Algorithm (GA) to solve the tool allocation problem in flexible manufacturing systems (FMS). Tool allocation on machines' tool magazines is a setup problem in an FMS and has significant effects on the overall system performance in terms of flexibility, efficiency and utilization. The tool allocation problem as defined in this study involves allocation of tools into the limited tool magazines of a set of machines in an FMS in order to produce a set of batches for part types that have route flexibilities. Consideration of the operation route flexibilities for parts and allowing splitting of part batches among alternative machines are important characteristics of this study which differentiate it from other work on tool allocation. The performance measure is selected as a multi-criteria objective that minimizes the weighted sum of unsatisfied demand of parts and the workload imbalance among machines. The mathematical programming model of the problem is presented. Since the exact solution procedures such as branch & bound are intractable for the tool allocation problem, a GA procedure is proposed. The chromosome structure and genetic operators are specifically designed. The developed GA procedure is tested on a set of problems. It is observed that the proposed approach provides satisfactory results in reasonable computation times when compared to exact solution algorithms.

ÖZET

ESNEK İMALAT SİSTEMLERİNDE KESİCİ TAKIMLARIN YERLEŞTİRİLMESİ PROBLEMİNE GENETİK ALGORİTMA YAKLAŞIMI

Bu tez, esnek imalat sistemlerinde kesici takımların yerleştirilmesi problemini çözmeye yönelik olarak bir Genetik Algoritma (GA) geliştirilmesine odaklanmıştır. Tezgâhların magazinlerine kesici takımların yerleştirilmesi, esnek imalat sistemlerinde bir kurulum problemidir ve sistem performansına esneklik, verimlilik ve kullanım oranı bakımından önemli etkileri vardır. Bu çalışmada tanımlanan kesici takımların yerleştirilmesi problemi, esnek imalat sisteminde rota esneklikleri olan farklı parça türlerine ait parça gruplarının üretilebilmesi amacıyla bir grup tezgâhın sınırlı kapasitede olan kesici takım magazinlerine kesici takımların yerleştirilmesidir. Parçaların operasyon rotası esnekliklerinin göz önünde bulundurulması ve parça gruplarının alternatif tezgâhlar arasında bölünmesine izin verilmesi bu çalışmayı kesici takım yerleştirilmesi üzerine yapılan diğer çalışmalardan ayırt eden önemli özelliklerdir. Performans ölçüsü, karşılanamayan parça talebinin ve tezgâhlar arasındaki iş yükü dengesizliğinin ağırlıklı ortalamasını minimize eden çok kriterli bir amaç fonksiyonu olarak belirlenmiştir. Problemin matematiksel programlama modeli sunulmuştur. Kesici takımların yerleştirilmesine yönelik dal-sınır yaklaşımı gibi kesin çözüm yöntemleri çok zor ve karmaşık olduğundan bir GA yaklaşımı önerilmiştir. Kromozomun yapısı ve genetik operatörler probleme özel olacak şekilde tasarlanmıştır. Geliştirilen GA prosedürü bir problem seti üzerinde denenmiştir. Kesin çözüm algoritmaları ile karşılaştırıldığında, önerilen yaklaşımın makul hesaplama sürelerinde tatmin edici sonuçlar sağladığı gözlenmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

at	Initial Available Processing Time
it	Idle Processing Time
pt	Processing Time
q	Number of Parts to be Produced
ta	Tool Availability
tsc	Tool Slot Capacity
tsr	Tool Slot Requirement

AFV	Average Fitness Value
ANOVA	Analysis of Variance
CNC	Computer Numerical Control
DF	Degrees of Freedom
FMS	Flexible Manufacturing System
G	Maximum Number of Generations
GA	Genetic Algorithm
IP	Improvement Period
LAN	Local Area Network
MIP	Mixed Integer Programming
MSE	Mean Square Error
N	Population Size
PI	Improvement Probability
PM	Mutation Probability
PTS	Part Type's Score
PX	Crossover Probability
RWS	Roulette Wheel Selection
SPT	Shortest Processing Time
SS	Sum of Squares
WAN	Wide Area Network
WFV	Worst Fitness Value

1. INTRODUCTION

Surviving in today's fast developing and competitive global markets depends on fulfilling customers' expectations. The performance of the product is not the only critical customer expectation. In addition to performance, customers demand variety of products in low costs, and in no time. Manufacturers are obliged to meet all these expectations simultaneously, in order to compete and survive in the market.

In the history of industrial development, a number of alternative production systems have been developed and operated to meet different types of customer demand. Job shop production systems support producing high variety of products and could be used meet the demand for customized products. However, due to their low production volumes, costs remain in high levels in such production systems. Mass production systems (or transfer lines), introduced in the early 20^{th} century, enable high volumes of production. Due to their increased efficiency levels, they can achieve much lower costs than job shop production. Although they seem to be superior, mass production systems are inflexible and cannot be used to serve for variety of products.

Flexible Manufacturing Systems (FMS) have been developed as an intermediary between job shop and mass production systems, and manufactures products in medium volumes and medium varieties. In its operations, an FMS aims to reach mass production's efficiency levels while retaining the flexibility of a job shop. FMS uses advanced and automated manufacturing technologies, integrated material handling systems and is managed by a computerized control system. The first FMS installations in the world have been in USA in 1960's.

The development in automated manufacturing technologies such as numerically controlled (NC) machine tools, material-handling systems, robotics and computer control systems have led to the current state of FMSs. The first successful NC machine was demonstrated in 1950s. The development of control system in 1971, led to micro-computer controlled NC machines, also known as CNCs. CNCs' main advantage was their ability to store various part programs in their memory and to communicate with the computer

control systems. Computer controlled material-handling systems decreased waiting time and work-in-progress inventory when compared to manual loading, unloading and manual material handling systems. The Robotics Industries Association defines an industrial robot as a programmable, multi-functional manipulator designed to move materials, parts, tools, or special devices through variable programmed motions for the performance of a variety of tasks. The integration of computer control systems such as management information systems, database management systems and shop floor control systems; advances in local and wide area networks (LANs and WANs), bar codes, programmable logic controllers and computer controls; automatic identification, data collection and analysis systems made great contributions to factory automation (Singh, 1996).

Flexibility of a manufacturing system can be defined as a collection of properties that support changes in production activities or capabilities (Carter, 1986). Changes can either be internal (breakdowns, software failures, etc.) or external (changes in product design, demand, product mix, etc.). In order to compensate for the effects of these changes, versatility of equipment is necessary.

The literature defines many different forms of flexibility in an FMS. *Machine Flexibility* in an FMS is the flexibility that machines and existing tools provide during the course of production. CNC machines perform operations such as machining, inspection, assembly and sheet metal presses on part families. When a part type arrives at a CNC, the CNC automatically chooses the appropriate tool from its tool magazine and performs the required operation. Therefore which tools are mounted on a CNC's tool magazine is a critical issue and it is considered as a setup problem in FMS planning cycle.

Route Flexibility refers to the ability of an FMS to manufacture or assemble parts along alternative routes. Due to the versatility of machines provided by the allocation of different tools, route flexibility allows for various route selection mechanisms of operations during the production of a product.

The tool allocation problem involves the allocation of tools and loading of operations among a group of machines, so as to maintain high machine and route flexibilities in the FMS. A proper tool allocation allows for efficient operation loading and in turn, results in

high production amounts and utilization levels in the FMS. This study aims at finding a best way to allocate tools to machines' available slots in their tool magazines, such that the unsatisfied demand of part types and workload imbalance among machines in an FMS are minimized within the planning horizon.

The mathematical programming model developed for the tool allocation problem in FMS is very hard to solve by using exact algorithms. The computation time required to solve the problem increase exponentially with increasing problem size. On the other hand, metaheuristics offer effective and efficient solution methods to find good solutions for problems where exact search algorithms are inapplicable. Genetic Algorithm (GA), which is a popular and widely used metaheuristic is applied in this study to search for good solutions for the tool allocation problem.

Section 2 contains the literature survey on mathematical modeling and heuristic approaches to the tool allocation and machine loading problems in the literature. In section 3, presents a detailed problem definition along with a mixed integer programming (MIP) model together with related assumptions. Section 4 proposes the GA approach as a solution methodology to the tool allocation problem. Numerical study and the results are presented in section 5. Conclusions are presented in section 6.

2. LITERATURE SURVEY

The basic objective of an FMS could be summarized as “to achieve the efficiency and utilization levels of mass production, while retaining the flexibility of manually operated job-shops” (Stecke and Solberg, 1981). Stecke defines the five production planning problems as follows (Stecke, 1983):

(1) Part type selection problem that deals with choosing a subset of part types to be processed immediately as well as simultaneously.
(2) Machine grouping problem that partitions the machines into functional groups with respect to operation capabilities to enhance overall performance.
(3) Production ratio problem that determines the relative ratios of the part types, namely the job sizes, of the subsets chosen in (1).
(4) Resource allocation problem that allocates the limited number of pallets and fixtures of each fixture type among the selected part types.
(5) Loading problem that allocates operations of part types and their corresponding tools to the machine groups subject to a number of constraints.

This study concentrates on the loading problem in an FMS. This problem has gained a wide interest in the literature and researches have been carried out related to the subject. Stecke listed several alternative loading objectives (Stecke, 1983):

- Balancing the assigned machine processing times.
- Minimizing the number of movements (transportation) from machine to machine.
- Balancing the workload per machine for a system of groups of pooled machines of equal sizes.
- Unbalancing the workload per machine for a system of groups of pooled machines of unequal sizes.
- Filling the tool magazines as densely as possible.
- Maximizing the sum of operation priorities.

The tool allocation problem in an FMS is mostly studied as a subproblem of the broader concept, the loading problem. The generic objective of the tool allocation problem may be defined as "choosing the right allocation of tools on tool magazines of the machines in an FMS such that the performance criteria are optimized". The limitations of the problem are machines' tool magazine capacities and the number of tool copies present in the system.

The tool allocation problem is closely related to many of the FMS problems and it is hardly ever considered independently. Even though it is a part of the design of FMS layout, the performance of this layout can only be measured by considering decisions, rules and procedures made during the operation of the system. So all design, planning and scheduling issues together with their subproblems may well be related to and treated together with the tool allocation problem.

Different approaches to the machine loading problem (and in turn, tool allocation) have been considered for almost three decades. These approaches include mathematical modeling, simulation based, multicriteria and decision making, and heuristic based methodologies.

Mathematical modeling and heuristic based methodologies are the two main research areas of this study. The literature survey has been focused on these two areas.

Loading in FMS is a broad concept and there is an extended literature presenting a variety of different approaches on the loading problem. The loading problem can be analyzed in three main categories (Grieco et al., 2001).

- The characteristics of the FMS
- The characteristics of the plant where the FMS operates
- The interfaces of the loading module with the upper and lower levels in the production hierarchy

The characteristics of the FMS involve the physical design and inner dynamics of the FMS that should be considered when formulating the loading problem. Types of machines,

controlling systems, tools and tool handling systems, parts, pallets and fixtures as a whole determine the capabilities of an FMS and are considered as the constraints of any mathematical modeling of a loading problem.

The characteristic of the plant where the FMS operates determines the production environment and the constraints that are set against the loading problem. Constraints like setup times, labor effects, maintenance of the machine tools and production volumes put limitations on the FMS and should be taken into account when modeling the FMS.

The position of loading and its interactions with higher and lower levels of planning is an important aspect. In other words, if the loading problem is a subproblem within the planning environment, the interactions are the inputs and outputs of this subproblem. Higher level planning modules (e.g. MRP, ERP) transmits requests to the loading module in an FMS. Handling of these requests, due dates and priorities affect the loading problem. Lower level planning includes managing unforeseen events like tool breakage and limitations in the management software like the software not supporting tool sharing policies among machines.

Grieco et al. (2001) group loading problems in an FMS into four main categories.

- Machine types (general or identical)
- Tool management policies (batching, flexible or hybrid)
- Objective functions
- Constraints

As opposed to the early FMS systems with general machines dedicated to different purposes (grinding, assembly, washing, etc.), machine types in new FMSs are less diversified and mostly consist of identical machines with broad machining capabilities.

Among the tool management policies, batching is the most common one. Batching involves making the tool allocations at the beginning of each planning period, and keeping that configuration of tool magazines unchanged till the end of that planning period. The flexible tool management policy allows for tool exchange among different machines and

tool storage within the planning period while the machines are working. Hybrid strategy puts some limitations on the flexible strategy such as setting preferred locations for some of the tools.

Various types of objective functions have been proposed and applied for loading problems in the literature. Optimizing costs (manufacturing, inventory, profit), flow-times, makespan, tardiness, production rates, workload balancing among machines, loading of particular subsystems (part transport, tool transport, refixturing), number of alternative routes and changes in tool magazine configurations.

Constraints of the loading problems are mostly related to loading of operations on machines, limitations on available machining times, production capacities, tool allocations on machines, tool magazine capacities, tool availabilities (number of tool copies from each tool type), number of pallets and fixtures, workload balancing on machines and due dates for part types.

Despite the wide applications of mathematical modeling in the literature on machine loading, the computation time required by these models remains prohibitive. Heuristics offer considerable computational simplicity and storage benefits, while retaining good solutions. This is the rationale behind the initiative on the development of a heuristic procedure on machine loading problem.

Shanker and Tzen (1985), Shanker and Srinivasulu (1989), Mukhopadhyay et al. (1992), Tiwari et al. (1997), Mukhopadhyay et al. (1998), Tiwari and Vidyarthi (2000), Vidyarthi and Tiwari (2001), Sarma et al. (2002), and Swarnkar and Tiwari (2004) build up the greater portion of the literature on developing heuristics for the machine loading problem. All the studies in this portion have common assumptions like unique part routing, non-splitting of the jobs. Certain operations can be processed on certain machines and all procedures have same or similar objectives.

Shanker and Tzen have studied the loading problem in a random FMS with objectives of balancing the workload and meeting the due dates, so as to minimize number of late jobs (Shanker and Tzen, 1985). The mixed integer nonlinear programming

formulations for the loading problem are very large and difficult to be solved, so they proposed two heuristics. The first heuristic aims to only balancing the workload among machines in the FMS. This is done by creating a descending route processing time list for all job-route pairs in the system and allocating the first job from the route processing time list and its related tool on the machine with the greatest remaining available processing time such that the tool magazine capacity constraint is not violated. The workload imbalance is calculated as the sum of absolute values of overload and underload on each machine. The second heuristic proposed intends to balance workload and minimize number of late jobs at the same time. The difference of this second heuristic is that jobs are classified with respect to their due dates and that the allocation order on machines is done with respect to this classification.

Shanker and Srinivasulu (1989) have developed heuristics similar to those developed by Shanker and Tzen. The heuristics differ in balancing the workload and the way jobs are selected. They define the critical resource (number of tools and processing time) and critical machine (the one with least/most remaining capacity) concepts to decide at each stage of assignment in their heuristic. Their objective is to minimize workload imbalance and to maximize throughput. Another loading heuristic with the same objective as that of Shanker and Srinivasulu is developed by Mukhopadhyay et al. (1992). Their heuristic gets the job sequence by SPT and uses an essentiality (uniqueness of a machine for an operation's allocation) ratio for loading jobs' operations to machines. Essentiality ratio (for part-operation-machine) is lower if an operation can be processed on alternative machines other than a single machine. Tiwari et al. (1997) proposed a heuristic which was a modification of the one developed by Mukhopadhyay et al. (1992), which resulted in improved performance.

Mukhopadhyay et al. (1998) and Tiwari and Vidyarthi (2000) incorporate metaheuristics in machine loading. The former employs Simulated Annealing (SA), while the latter uses Genetic Algorithm (GA) in finding the most appropriate input part sequence. Their solution representations are strings of characters that correspond to part types and show processing sequence of part types in the FMS. Their SA algorithm uses a perturbation method, while their GA uses various crossover and mutation methods on the solution representation to find the best input part sequence. They both use the heuristic

method developed by Mukhopadhyay et al. (1992) to make the tool allocation and to calculate the system imbalance objective. Vidyarthi and Tiwari (2001) have the same system imbalance objective, but introduce a fuzzy-based heuristic to the input part sequencing. Their heuristic gives decisions based on a number of membership functions for the input part sequencing and operation allocation on machines.

Sarma et al. (2002) use the objective of minimizing system imbalance and maximizing throughput with tool slot and machine availability constraints. They propose two different heuristics, one of them taking the input part sequence with predetermined rules and the other with Tabu Search (TS) based heuristic. The objective values of these different input part sequences are calculated with a heuristic which is a procedure allocating tools and operations on machines such that the resulting configuration is feasible.

Swarnkar and Tiwari (2004) implement a hybrid approach: TS for input part sequence, combined with SA for the allocation of operations on machines. Objective function value is calculated by the method presented by Tiwari et al. (1997).

Table 2.1 represents a classification of some of the loading heuristic approaches in the literature. In most of the studies displayed in Table 2.1 the procedure developed by the authors has been compared to the former procedures on the subject. None of them are compared to optimal results. Evaluating the performance with respect to the optimal solution will be one of the main focus areas of this study.

One should also note that, all the heuristics mentioned in Table 2.1 have the common characteristic of allocating tools on machines after the input part sequence is determined and many other decisions are made. Tool allocations are done as the results of the heuristic procedures.

Table 2.1. A classification of loading heuristics in the literature

Study	Problem	Performance (in terms of objective function value)
Shanker and Tzen, 1985	8 part types, each with 1-3 operations, 4 machines	Improved over sequential loading of operations
Shanker and Srinivasulu, 1989	Problem of Shanker and Tzen, 1985	Improved over Shanker and Tzen, 1985
Mukhopadhyay et al., 1992	Problem of Shanker and Tzen, 1985 and 9 more defined and tested	Improved over Shanker and Srinivasulu, 1989
Tiwari et al., 1997	Problems of Mukhopadhyay et al., 1992	Cannot say to have improved over Shanker and Srinivasulu, 1989 and Mukhopadhyay et al., 1992
Mukhopadhyay et al., 1998	10 part types, each with 1-3 operations, 6 machines	No comparison presented
Tiwari and Vidyarthi, 2000	Problems of Mukhopadhyay et al., 1992	Improved over Tiwari et al., 1997
Vidyarthi and Tiwari, 2001	Problem of Shanker and Tzen, 1985	Improved over Mukhopadhyay et al., 1992 in most of the cases
Sarma et al., 2002	7 part types, each with 1-3 operations, 4 machines	No comparison presented
Swarnkar and Tiwari, 2004	Problems of Mukhopadhyay et al., 1992	Improved over Mukhopadhyay et al., 1992 in most of the cases

Kumar and Shanker (2000) propose a GA to solve input part sequence and machine loading. They define three types of decision variables all of which are binary variables. x_p denotes whether a part type p is selected or not, y_{mt} denotes if tool t is allocated to machine m, z_{mt}^{po} denotes if a part-operation pair *(p,o)* is assigned to the process alternative *(m,t)*. Their solution is a string made up of these binary variables. They further improve the solution representation by reducing the length of the chromosome by denoting the

processing alternative by an integer for z_{mt}^{po}. Using the binary variables of the solution, they calculate constraints like machine workload; number of tools used and tool magazine capacities. Their objective function is the number of these constraints of the MIP violated by that solution representation subtracted from the number of part types selected. They generate 10 problems and test their procedure with CPLEX. GA resulted in optimal or near optimal solutions.

Arikan and Erol (2006) focused on part type selection and introduced the matrix based machine tool allocation representation as the solution encoding of the SA and TS algorithms that they have developed. If tool t is allocated on machine m, the corresponding binary variable is set to 1. The generation of new solutions is done by add and drop moves which correspond to changing a 0 value to 1 and a 1 to 0. Their objective is to maximize the weighted sum of selected part types. They use a subroutine to calculate objective value based on the move type. They have compared their results with the optimal GAMS solutions and attained optimal and near optimal results.

Here, it is noticeable that both Kumar and Shanker (2000) and Arikan and Erol (2006) make the tool allocation first. The calculation of the corresponding fitness value is done afterwards.

3. A MATHEMATICAL PROGRAMMING MODEL FOR THE TOOL ALLOCATION PROBLEM IN FMS

FMS applies various processes on different part types to manufacture products or sub-products for the customers. The demand for a part type determines the quantity of parts from that part type to be produced within the planning period. The part types and their quantities form up the part-mix of the problem.

Each part type requires a number of operations to be performed in a predetermined order. Every part type in the system must follow the operations in order to become the end-product. This order (or the sequence) of operations is called that part type's route.

Operations are done by machines in the FMS. Machines can process various types of operations of different part types. Machines' capability to process various operations is determined by the tools that are loaded on the tool slots of their tool magazines. Whenever an operation is assigned to a machine, it automatically acquires the appropriate tool from its tool magazine.

The production capacity and efficiency measures of an FMS are highly affected by the allocation of tools on machines. A good loading aids an FMS to make production in an effective and efficient manner. Unwise allocation of tools may cause underutilized tools and unused available processing times for machines both of which reduce system's overall performance.

The tool allocation problem in an FMS aims to allocate tools on machines in such a fashion that the FMS performance is optimized in terms of the objective(s).

One may easily come up with the idea of loading all types of tools to every machine so as to neutralize the effect of tool allocation. However, almost always, tool availabilities and the slot capacities of machines in the system are limited. This rises up the question of how to make the tool allocation in an FMS.

3.1. Model Assumptions

Route Flexibility involves various selection mechanisms of operations during the production of a product. Browne et al. (1984) defines the three types of route flexibilities as follows:

- *Processing Flexibility* is determined by the possibility of producing the same manufacturing feature with alternative operation sequences
- *Sequencing Flexibility* refers to the possibility of interchanging the sequence, in which required manufacturing operations are performed
- *Operation Flexibility* is the capability of processing an operation in different machines

This study assumes all these three types of route flexibilities. In other words, each part type in the FMS is allowed to follow alternative routes, where each operation of each route can be processed on alternative machines. Routes that are formed by interchanging the sequence of operations are thought as additional alternative routes to the part type's set of alternative routes.

Route flexibility assumptions can easily be observed by implementing the network representation of each part type's alternative routes. Processing flexibility can be observed given that the network satisfies the below conditions.

- Every node in the network represents the part type's operations
- The network is acyclic
- The network has a single source node and a single sink node that correspond to receival and shipment operations of the part type.

However, such a network would only reveal the processing flexibility of the problem OprRA and OprSA stands for receival and shipment operations and are treated as operations without processing times and do not need to be loaded on any machine. Figure 3.1 is an illustration of processing flexibility, where the part type can follow two alternative processing sequences.

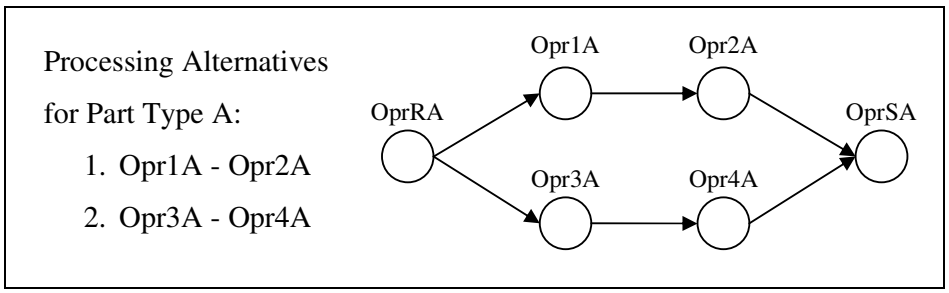


Figure 3.1. Processing flexibility in FMS

The sequencing flexibility can be shown as additional routes to the original network processing flexibilities. It is not possible to represent the sequential flexibility of the model by using an acyclic network. For this reason dummy nodes will be added to the network representation standing for the interchangeable operations. For instance, if operations Opr3A and Opr4A in the second route are interchangeable, dummy nodes Opr3A and Opr4A can be added to the network and the sequence Opr4A-Opr3A can be treated as a completely different set of operations although they are not (Figure 3.2). The resulting route becomes the third alternative route without loss of generality.

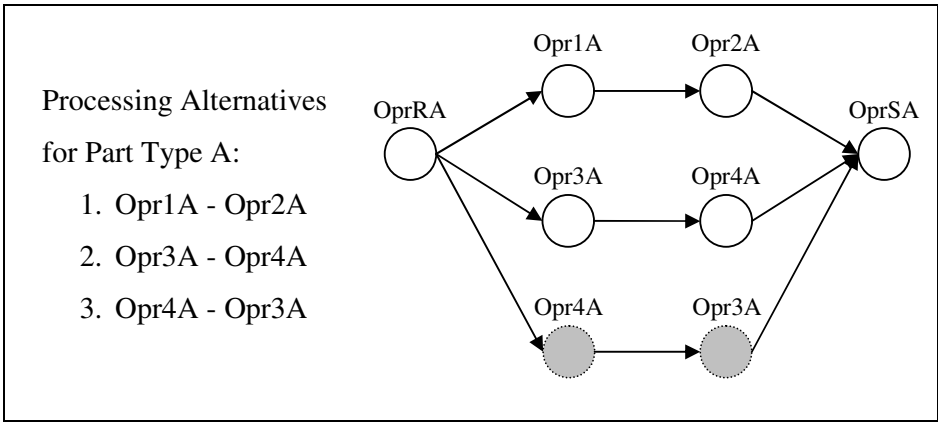


Figure 3.2. Sequencing flexibility in FMS

Operation flexibilities asserting the possibility for processing each route's each operation on alternative machines, increases the number of alternative routes for a part type in the FMS. For instance, in the situation where each of Opr1A and Opr2A of Figure 3.2 could be processed by two alternative machines (Opr1A on m_1 and m_2; Opr2A on m_3 and

m_4), and Opr3A and Opr4A could be processed by single machines (Opr3A on m_3; Opr4A on m_4). In this case, all alternative routes for A could follow is shown in Table 3.1.

Table 3.1. Alternative routes in FMS

All Alternative Routes for Part Type A:
1. Opr1A(m_1) - Opr2A(m_3)
2. Opr1A(m_2) - Opr2A(m_3)
3. Opr1A(m_1) - Opr2A(m_4)
4. Opr1A(m_2) - Opr2A(m_4)
5. Opr3A(m_3) - Opr4A(m_4)
6. Opr4A(m_4) - Opr3A(m_3)

The part types that are to be processed by the FMS arrive in batches. The batch of a part type is allowed to be split and it is legal to meet the demand of a part type partially. Besides, loading of operations on machines is done one by one meaning that each part may use a different alternative route to complete its operations. The processing of every operation requires a predetermined processing time. The model assumes no preemption, which prohibits the interruption of a part type's operation once it is started.

Machine flexibility is offered by tools and the machines that the tools are allocated on. The machines in the FMS are assumed to be identical with each other, meaning that that all part types and their operations can be processed in all machines in equal processing times, given that the tool related to that operation is loaded on the machines. Available processing times and slot capacities are equal for all machines.

The tool allocation is done at the beginning of the planning period and remains unchanged throughout the period. When the period ends, the tool allocation on tool magazines can be modified if needed.

The tool commonality assumption is valid for the tool allocation model. Each operation in the FMS can be processed by a unique tool, whereas different operations may

share the same tool. Tools sizes are not equal for each tool type. However tool copies are identical in size and functionality.

Tool-machine compatibility assumption guarantees the possibility of allocation of all tools on all machines as long as the slot capacities are not exceeded.

3.2. MIP Formulation of the Tool Allocation Problem

This section introduces mixed integer programming formulation (Albey, 2006) for the above described tool allocation problem. Table 3.2 depicts the notation used in the formulation.

Table 3.2. Notation of the MIP formulation for tool allocation problem

Indices

Part types $i = 1, 2, \ldots, I$

Machines $m = 1, 2, \ldots, M$

Operations $o = 1, 2, \ldots, O$

Tools $k = 1, 2, \ldots, K$

Parameters:

tsc_m : Tool Slot Capacity for machine m.

tsr_k : Tool Slot Requirement for tool k.

q_i : Number of parts to be produced from part type i.

ta_k : Tool availability for tool k.

at : Initial Available Processing Time for machines.

it_m : Idle Processing Time for machine m.

pt_{om} : Processing Time of operation o on machine m.

z_{ok} : Equal to 1 if tool k is required for operation o, o/w zero.

Table 3.2 continued

Variables:

D_{km}	: Equal to 1 if tool k is allocated on machine m's tool magazine, o/w 0.
X_{om}	: Amount of operation o processed on machine m.
$U^{+}_{mm'}$, $U^{-}_{mm'}$	: Differences between two machines' utilizations.
$F_{oo'}$	: Amount of operations flowing from o to o'.
S_i	: Unsatisfied demand of part type i.

Sets:

PAIRS(m)	: All possible machine pairs in the system (m,m') st. m≠m'
LO(i)	: Last operation (shipment) of part type i
S(o), P(o)	: Successors and Predecessors of operation o.
T(o)	: Tool required for processing of operation o.

The multi-criteria objective determined for the tool allocation problem in this study is minimizing the weighted sum of unsatisfied demand of parts and the workload imbalance among machines (Equation 3.1).

Unsatisfied demand is calculated by summing up the completion ratios of each part type. This sum is weighted by the initial number of parts of the part type that has the maximum demand.

The second part of the objective function, (adopted from Kumar and Shanker, 2001) calculates the average of the pair wise differences between each machine's utilizations and aims to balance the workloads assigned to each machine. The complete formulation is presented as follows:

$$Z = \min\left\{ \max_i\{q_i\} \times \left[\sum_i \left(\frac{S_i}{q_i}\right) \right] + \left[\frac{\sum_{\forall (m,m') \in PAIRS(m)} \left(U^{+}_{mm'} + U^{-}_{mm'} \right)}{\frac{Card(m) \times \left[Card(m) - 1 \right]}{2}} \right] \right\} \tag{3.1}$$

st. $$\sum_{\forall k} tsr_k \times D_{km} \le tsc_m, \ \forall m \tag{3.2}$$

$$\sum_{\forall m} D_{km} \le ta_k, \ \forall k \tag{3.3}$$

$$\left(\sum_{\forall o} pt_{om} \times X_{om} \right) \le at, \ \forall m \tag{3.4}$$

$$X_{om} \le \max_i\{q_i\} \times \sum_{k \in T(o)} \left(z_{ok} \times D_{km} \right), \ \forall o, \ \forall m \tag{3.5}$$

$$\sum_m X_{om} = \sum_{o' \in S(o)} F_{oo'}, \ \forall o \tag{3.6}$$

$$\sum_{o \in P(o')} F_{oo'} = \sum_m X_{o'm}, \ \forall o' \tag{3.7}$$

$$U^{+}_{mm'} - U^{-}_{mm'} = \frac{\left(\sum_{\forall o} pt_{om} \times X_{om} \right) - \left(\sum_{\forall o} pt_{om'} \times X_{om'} \right)}{at}, \ \forall (m,m') \in PAIRS(m) \tag{3.8}$$

$$\sum_{o \in LO(i)} \sum_m X_{om} + S_i = q_i, \ \forall i \tag{3.9}$$

$$D_{km} = \{0,1\} \tag{3.10}$$

$$X_{om}, \ F_{oo'} \ \textit{are integer} \tag{3.11}$$

$$U^{+}_{mm'}, \ U^{-}_{mm'}, \ S_i \ge 0 \tag{3.12}$$

Equation (3.2) is the machine slot capacity constraint and it assures that the total amount of slot requirements of the tools allocated to each machine cannot exceed the total slot capacities of machines.

Equation (3.3) is the tool availability constraint and it asserts that the tool number of tool copies that can be mounted on machines is limited by the number of that tool's availability.

Equation (3.4) is the machine availability constraint and it ensures that the total processing times of the loaded operations on a machine cannot be larger than the total available processing time of that machine.

Equation (3.5) is the producibility constraint for loading operations on machines. Given that an operation's required tool is already allocated on that machine, it defines the maximum demand in the part mix as the upper bound to the number of operations that can be loaded to that machine. If the required tool is not allocated on the machine, RHS is zero and operation is not loaded on machine. If the tool is allocated the maximum number of operation allocated to the machine is bounded by the maximum demand.

Equation (3.6) and (3.7) are network flow constraints. These constraints provide the complete flow of part types throughout the FMS. Alternative routes let an operation to have multiple successors and predecessors. Besides, an operation can be processed in alternative machines in the FMS. In order to guarantee the completion of each part type, the total flow of a specific operation and the total flow of its successor operations are made equal.

Equation (3.8) is the definition for the absolute pair wise differences of machines' utilizations those are used in the objective function in order to balance the workload among machines.

Equation (3.9) is the definition representing the quantity of unsatisfied demand for each part type.

3.3. Problem Complexity

One of the two subproblems of the whole problem, the tool allocation on machines, denoted by constraints (3.2) and (3.3), is equivalent to the bounded knapsack problem which is proven to be NP-Hard (Martello and Paolo, 1990). For this reason, the tool allocation problem defined in section 3.2 is NP-Hard.

4. THE PROPOSED GA APPROACH TO THE TOOL ALLOCATION PROBLEM

Due to the complexity of the model developed for the tool allocation problem, alternative solving procedures are considered. Genetic Algorithms (GA), offer good solutions for hard problems in reasonable computation times.

GAs are based on the adaptive processes that are observed in natural systems. Natural systems tend to adapt themselves to changes over time. This is achieved through the changes that occur in organisms' genes. Genetics is the branch of science that studies genes. Genes of an organism contains the necessary information that is used to build up the structures and carry out the operations of organisms.

Heredity and variations in the genes form the basis of genetics. These two components allow the organisms, or more generally the populations, to adapt to changes over time. Population genetics, which is based on classical genetics, studies the distribution of and change in genes. Heredity is the information that is passed to young organisms from their parents. The adapted characteristics of parents are preserved in the youngsters by heredity. On the other hand, genes in natural systems tend to show variation over time. This phenomenon is due to occurrences like crossover and mutation, which in fact result in alterations in the genes. These alterations in genes may be beneficial, ineffective or harmful. It is the beneficial alterations, which allow the organisms, and in turn natural systems to adapt. The adapted individuals survive, while the ones that do not adapt diminish and finally, become extinct. This natural phenomenon is called natural selection.

The "fit" individuals in the population, those having advantageous genetic structure, live longer than the others and have greater chance of reproduction. The chromosomes of the fit individuals are passed to next generations by means of reproduction. One would expect that this phenomena cause the population to intensify and form a population made of individuals very similar genetic structures. However, alterations in the genes that occur by crossover and mutation cause the chromosomes, in turn the individuals, to differ and retain the population's genetic diversification.

GA has been developed by John Holland and his research team. GA is an an analogy of the natural systems. It is a search algorithm that builds up a population and imitates the operation mechanism of adaptation to come up with a combination of genes that produces the best, in scientific terms, the “fittest” organism.

GAs and other random search techniques have become quite popular in recent years. This is due to the fact that enumerative and calculus-based search techniques have obvious shortcomings when compared to random search techniques. Factors that contribute to GA’s robustness and make it advantageous over other traditional methods are summarized by Goldberg (1989):

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

The formulated MIP model for the tool allocation problem also involves a set of parameters and GA will be applied to find a combination of these parameters so as to find a good solution.

The MIP formulation involves finding an optimal combination of variables for tool-machine allocations, choosing routes to flow the parts through and loading of operations on machines.

The developed GA will search for a good solution by recombining and altering the individuals that represent alternative tool-machine allocations. The fitness of an individual will be calculated by choosing among alternative routes and loading these routes’ operations on machines in the FMS.

In other words, the individual will address a planning problem, whereas the measure of fitness will deal with scheduling issues of the FMS.

The main concerns in a GA are:

- The Solution Representation
- Generation of the Initial Population
- Fitness Evaluation of a Solution
- Selection
- Crossover
- Mutation
- Advancing the Population to the next Generation

For a GA application, all of the above concerns need to be studied and developed in order to match the tool allocation problem. These concerns build up the foundations of GA and have essential effects on the performance.

There are many different GA approaches and applications in the literature. The generic structure of the proposed GA is presented in Table 4.1.

Table 4.1. Pseudo code for the proposed GA

(1) ***Generate the initial population***
(2) ***Evaluate the fitness values*** *of the individuals in the population*
(3) If maximum number of generations is reached, STOP
(4) Preserve the best solution in the population and calculate the selection probabilities
(5) While number of children produced is less than population size
  (5.1) Select two mates from the population
  (5.2) ***Apply crossover*** *on two mates to form two children*
  (5.3) ***Apply mutation*** *on each child*
  (5.4) ***Improve*** *children if improvement conditions are satisfied*
(6) ***Advance the population*** *to the next generation's population, and go to (2)*

All the above generic steps of the algorithm must be adapted so as to meet the requirements and structure of the tool allocation problem in order to come up with a well-developed methodology giving high performance.

4.1. The Solution Representation

Holland (1992) defines an organism as an amalgam of characteristics determined by the genes in its chromosomes. Genes have a number of alternative forms, called alleles, each of which produces different characteristics in the organism.

Making an analogy of natural systems, in GA, each solution (individual) has a representation (chromosome) made up of numbers or characters (genes). The various combinations of these genes of different alleles can produce different structures (genotypes) all of which can be seen as a different solution.

The solution representation in a GA implementation is usually a user-defined string of numbers or characters, which are mostly problem specific. A proper solution representation is important and contributes to the effectiveness and efficiency of the implementation.

In most GA applications, the natural parameter set is coded as a string of characters of definite length. Various genetic operations are applied on this string of characters to find an optimum combination giving the best fitness value.

As it is presented in the model, the tool allocation problem involves a set of binary variables, each of which denotes an assignment of a tool to a machine. The chromosome designed for this specific problem is a matrix of binary variables. The matrix-based solution representation is advantageous in several aspects. When the solution matrix is observed, it is very easy to understand which tool is allocated on which machine. Besides, the two feasibility conditions of number of available tool copies in the system and tool magazine capacity for machines for the solution can be checked easily with the matrix representation.

Consider an FMS with tools 1 to k and machines 1 to m. The typical solution representation for this system is a k×m matrix. Each cell, denoted by (i, j) where i = 1, 2, …, k and j = 1, 2, …, m corresponds to tool-machine assignment and has a binary value. If tool i is allocated on machine j, then cell (i, j) will be 1, otherwise 0.

Each column corresponds to a machine's tool magazine (e.g. m_1, m_2, etc.) and each row corresponds to the tool copies of a specific tool (e.g. t_1, t_2, etc.) For a feasible solution, column sums are constrained by the maximum capacities of the tool magazines of machines. Besides, row sums for a solution must be less than or equal to total number of tool copies present in the system.

Table 4.2. The solution representation of an individual

	m_1	m_2	m_3
t_1	1	1	0
t_2	0	1	1
t_3	1	0	1
t_4	0	0	1
t_5	0	1	0
t_6	1	0	0

The two sets of constraints (3.2) and (3.3) of the MIP model can easily be checked by using the representation depicted in Table 4.2. Columns together with the tool slot requirements, give the machine slot capacities information, whereas each row sum denotes the number of tools available from the corresponding tool present in the system:

- $\sum_{\forall k} tsr_k \times D_{km} \le tsc_m, \ \forall m$ ⇔ Columns of the solution representation (4.1)
- $\sum_{\forall m} D_{km} \le ta_k, \ \forall k$ ⇔ Rows of the solution representation (4.2)

The solution representation in Table 4.2 also avoids tool duplications on machines. The allocation of a specific tool, more than once on the same machine is not allowed and makes no sense for this problem under the current assumptions.

One major guideline of the GA developed in the study is that each solution in the population is different from each other. This is called "No duplicates strategy". It is

thought that by applying this approach, the diversification of the population would be maintained through generations.

Assuming that the machines are identical, one can easily see that a typical FMS layout can be represented with alternative matrices. In fact, the number of alternative matrices that correspond to a single specific layout can be as high as $P(m,m) = m!$ (this number is less when all columns are not unique). This is due to the fact that columns in the representation are interchangeable and when interchanged, the matrix corresponds to an equivalent FMS configuration. For this reason, a procedure is developed to identify and combine equivalent solutions under one unique label. Since every cell in the matrix is binary, a binary coding procedure is applied for solutions. The labeling procedure for chromosomes (solutions) is depicted in Table 4.3.

Table 4.3. Pseudo code for labeling the chromosomes

(1) For every machine, a value corresponding to each machine's tool configuration is determined (There are k tools and m machines in the FMS and binary values at each cell (i,j) of the matrix represents whether i^{th} tool is allocated to j^{th} machine).

Set i = 1 and j = 1
While (j is less than or equal to m)
{ $Value_j$ = 0, Power = 0
While (i is less than or equal to k)
{ If (i,j) is equal to 1, then Coefficient = 2^{Power} and increment m_j's value by coefficient
Power=Power+1, i= i+1
}
j = j +1
}

(2) The values calculated for all machines in (1) are ranked in nonincreasing order and "-" signs are inserted among them.

Using this procedure, equivalent chromosomes are labeled and duplicates of a chromosome can easily be tracked. Two duplicate individuals are not allowed in the population.

Consider an FMS with 3 machines (all with tool magazine capacities of 3) and 5 tools (each tool has 2 copies). Table 4.4 illustrates the operation mechanism of the labeling procedure.

(1) Values corresponding to the machines' tool configurations are calculated:

Table 4.4. Illustration of labeling procedure

Power	Coefficient		m_1	m_2	m_3
0	$2^0 = 1$	t_1	1	1	0
1	$2^1 = 2$	t_2	0	1	1
2	$2^2 = 4$	t_3	1	0	1
3	$2^3 = 8$	t_4	0	0	0
4	$2^4 = 16$	t_5	0	1	0
		m_j's value	5 (=1+4)	19 (=1+2+16)	6 (=2+4)

(2) Nonincreasing ordering of the machine values results in the label "19-6-5". This string represents the set of all 6 (=3!) configurations including "19-5-6", "6-19-5", "6-5-19", "5-6-19" and "5-19-6" as well as itself.

4.2. Initial Population

The initial population in a GA application is important in many aspects. In GA applications, it is always preferred to generate the initial population randomly. This enables a random distribution of the individuals in the search space of the problem.

Population size on the other hand, usually depends on the nature of the problem. Reeves and Rowe (2003) denotes that the underlying idea of the population size is a trade-off between efficiency and effectiveness. As the population size decreases, the chances for exploring the search space effectively also decrease. However if the population size is too large, the efficiency of the application decreases due to the increased computation time.

For this study, population size is chosen to be related to the size of the problem, defined by the number of tools and the number of machines in the FMS. Table 4.5 summarizes the generation of the initial population.

Table 4.5. Pseudo code for initial population generation

(1) ***Generate the initial population***
  (1.1) While number of individuals produced is less than population size
    (1.1.1) Generate an individual randomly
    (1.1.2) Label the individual
    (1.1.3) Insert the individual to the initial population
…

As remarked in section 4.1, a solution (chromosome), which denotes a tool-machine allocation configuration, is represented by a matrix. Moreover, this matrix which is composed of binary variables has some certain characteristics that enable it to embody the feasibility constraints in the FMS. A procedure for generating random matrices those obey the feasibility conditions seems necessary in this respect.

The procedure for randomly generating the solutions for the initial population is not straightforward as filling up the cells with arbitrarily chosen binary variables. The two feasibility constraints (machines' tool capacities and tool availabilities) lead to a more systematic approach for generating solutions.

One should also note that it is advantageous to have matrices be filled with 1's as dense as possible such that there are no slacks in the machine slot capacity and tool availability constraints. For instance, say, two solutions S1 and S2, are identical except for one cell *(i,j)*, which is 1 in S1 and 0 in S2. When comparing the quality of these two solutions, one can easily see that S1 is at least as good as S2, due to its superior flexibility, given by having tool i loaded on machine j. Presence of tool i on machine j, when compared to its absence, may lead to a better operation loading.

Solution generation procedure developed for this study enables complete allocation of tools on machines. For this reason, the chromosome generation procedure is expected to increase the chances of having good solutions in the initial population. Once a solution matrix (a chromosome) is generated, it is labeled as described in section 4.1 and inserted into the initial population. Generation of problem layouts which is described in section 5.1 has been done so as to enable this complete allocation. The procedure given in Table 4.6 as follows:

Table 4.6. Pseudo code for chromosome generation for initial population in GA

Ranked List of Tools *is a list of tools denoting the allocation sequence of tool types, where tools are ordered in nonincreasing order of their sizes.*
List of Machines *is a list of machines, order by machines' indices.*
Current Sum of Tool Sizes *is equal to the sum of the tool sizes of tool types those are not yet allocated in the procedure.*
Set of Available Machines *for a tool is the set of eligible machines, on which a specific tool under consideration can be mounted.*

(1) Rank all the tool types in nonincreasing order of their sizes and list them in Ranked List of Tools (for tool types of equal tool size, break ties arbitrarily)
(2) If there are any tools in the Ranked List of Tools, choose the top-ranked tool type and discard it from the list and go to (3), else STOP
(3) For all the machines in the List of Machines:
(3.1) If machine's available slot capacity is equal to Current Sum of Tool Sizes, allocate the current tool type on current machine
(3.2) If machine's available slot capacity is greater than current tool type's tool size, insert the current machine to Set of Available Machines for the current tool type, else prohibit the allocation of the current tool on the current machine

Table 4.6 continued

(4) If there are any tools remain unallocated from the chosen tool type, go to (4.1), else go to (2)

(4.1) Choose a machine randomly from Set of Available Machines for the current tool type

(4.2) Allocate the current tool type on the chosen machine and discard the machine from Set of Available Machines for the current tool type

The reason for ranking the tool types in their nonincreasing is illustrated as follows. Table 4.7 demonstrates FMS Layout 1 and the operation of the proposed chromosome generation procedure. All machines have three available slots.

Table 4.7. Means of ranking tool types with respect to tool sizes

Tool Sizes		m_1	m_2	m_3	m_4
2	t_1				
1	t_2	0	1	1	0
1	t_3	1	0	0	1
1	t_4	1	0	1	0
1	t_5	0	1	0	1

Suppose that ranking is not considered and the allocation sequence for the tool types is chosen arbitrarily as t_2-t_4-t_5-t_3-t_1. The first four tool types of the allocation sequence may result in the allocation shown in Table 4.7. In this case t_1, which is of tool size equal to 2, cannot be allocated on any of the machines since all machines have single slot capacities. In order to avoid such situations, tool types are ranked in the chromosome generation procedure. As a result of the ranking application, it is expected that the chromosomes generated by the procedure would not follow a uniform distribution. More important than that The above procedure is tested for its randomness in terms of its capability to generate all feasible solutions for specific problems. Table 4.8 shows the two different FMS layouts generated to check the capability of the generation procedure. The first of these layouts' tool types have unequal tool sizes, whereas the second's tool types have equal tool sizes.

Table 4.8. FMS layouts generated to check the generation procedure's capability

	Number of Machines	Slot Capacity	Types of Tools	Each Tool's Availability	Tool Sizes
FMS Layout 1	4 (m_1 to m_4)	3	5 (t_1 to t_5)	2 from each tool	4 of the tools have size 1, 1 tool have size 2
FMS Layout 2	3 (m_1 to m_3)	4	6 (t_1 to t_6)	2 from each tool	All tools have size 1

The set of all feasible solutions for these two FMS layouts, which are relatively small in size are generated by hand. The resulting set is compared with that of the procedure outputs. The results reveal that all feasible solutions can be generated by the generation procedure if sufficient number of trials is made. Table 4.9 and Table 4.10 show the solutions generated by the procedure after running the procedure for a million times for FMS Layout 1 and FMS Layout 2.

Table 4.9. Distribution of solutions for FMS layout 1

FMS Layout 2			
Chromosome Representation	% of Total	Chromosome Representation	% of Total
17-17-14-14	5.1%	26-26-5-5	5.2%
22-17-14-9	13.1%	28-17-14-3	13.2%
22-22-9-9	5.0%	28-22-9-3	13.2%
26-17-14-5	13.3%	28-26-5-3	13.3%
26-22-9-5	13.5%	28-28-3-3	5.1%

The variance in the percentages of the generated chromosomes in the first FMS Layout is due to the ranking of the tool types with respect to their tool sizes.

Being able to generate all possible feasible solutions in both cases, confirms the validity of the chromosome generation procedure. The procedure will be used to generate the initial population of the GA.

Table 4.10. Distribution of solutions for FMS layout 2

FMS Layout 2			
Chromosome Representation	% of Total	Chromosome Representation	% of Total
51-45-30	6.8%	57-54-15	6.8%
51-46-29	6.4%	58-39-29	6.4%
53-43-30	6.7%	58-45-23	6.4%
53-46-27	7.0%	58-53-15	6.8%
54-43-29	6.7%	60-39-27	6.5%
54-45-27	7.2%	60-43-23	7.0%
57-39-30	6.3%	60-51-15	6.5%
57-46-23	6.6%		

4.3. Fitness Evaluation of a Solution

The fitness of an individual can be viewed as a measure of the genotype's influence upon the future (Holland, 1992). Survival-of-the-fittest concept requires a classification of the organisms in the population with respect to their fitness. Every organism in a population has a fitness value and the degree, or the level of this fitness determines the future of both that organism and its children, and in turn the future of the whole population.

In GA, every solution representation corresponds to a fitness value. The procedure to determine this fitness value may require variable amounts of effort. It may be as easy as making a few straightforward operations, or as complex as executing an algorithm to calculate a fitness value.

In order to evaluate the fitness of a solution some measure of performance is to be developed. The performance of a tool allocation in an FMS can be assessed in a number of ways. The key performance measures for this study are chosen to be:

- The throughput (number of parts completed)
- The workload balance among machines

Given the tool-machine allocation ($D_{km},\ \forall k,m$) the mathematical model of section 3 reduces to the following model, and the exact solution of this model could return a fitness value. However, it is not computationally feasible to solve the model using an exact algorithm for every chromosome generated in GA.

$$Z \ = \ \min\left\{ \ \max_i\{q_i\}\times\left[\sum_i\left(\frac{S_i}{q_i}\right)\right]+\left[\frac{\sum_{\forall(m,m')\in PAIRS(m)}\left(U^{+}_{mm'} \ + \ U^{-}_{mm'}\right)}{\frac{Card(m)\times\left[Card(m) \ - \ 1\right]}{2}}\right]\right\} \tag{4.3}$$

$$st. \qquad \left(\sum_{\forall o} pt_{om}\times X_{om}\right)\le at\ ,\ \forall m \tag{4.4}$$

$$X_{om}\le \max_i\{q_i\}\times\sum_{k\in T(o)}\left(z_{ok}\times D_{km}\right)\ ,\ \forall o,\ \forall m \tag{4.5}$$

$$\sum_m X_{om}=\sum_{o'\in S(o)} F_{oo'}\ ,\ \forall o \tag{4.6}$$

$$\sum_{o\in P(o')} F_{oo'}=\sum_m X_{o'm}\ ,\ \forall o' \tag{4.7}$$

$$U^{+}_{mm'}-U^{-}_{mm'}=\frac{\left(\sum_{\forall o} pt_{om}\times X_{om}\right)-\left(\sum_{\forall o} pt_{om'}\times X_{om'}\right)}{at},\ \forall(m,m')\in PAIRS(m) \tag{4.8}$$

$$\sum_{o\in LO(i)}\sum_m X_{om}+\ S_i\ =\ q_i\ ,\ \forall i \tag{4.9}$$

$$X_{om},\ F_{oo'}\ are\ integer \tag{4.10}$$

$$U^{+}_{mm'},\ U^{-}_{mm'},\ S_i\ge 0 \tag{4.11}$$

Therefore, a subroutine that converts the tool allocation into the above measures is necessary. The design and development of this subroutine is a key issue in this study. The subroutine will be a surrogate for this operation loading model. The idea of using heuristics, which achieve good solutions in reasonable run times for complex problems, has been widely practiced in the literature.

The tool allocation, in other words the solution, is taken as input of the operation loading subroutine. The subroutine loads the part types' operations in accordance with their routing information, machines' availability and tool configurations. At each step of allocation, the available processing times of machines are updated (decreased) by the amount of workload that corresponds to each operation. The fitness evaluation procedure of the GA is presented in Table 4.11.

Table 4.11. Pseudo code for fitness evaluation in GA

…
(2) ***Evaluate the fitness values*** *of the individuals in the population*
    (2.1) ***Evaluate the fitness values*** *of the individuals by using the operation loading heuristic algorithm*
    (2.2) *If the total number of individuals is greater than the population size, destroy the individual with the worst fitness value (break ties arbitrarily)*
(3) *If maximum number of generations is reached, STOP*
…

The two kinds of information extracted from the results of the heuristic (unsatisfied demand of parts and idle times of machines after the allocation) enable the determination of the fitness value corresponding to any given chromosome.

The children are produced in pairs and if a child produced is identical to a previously produced child in the same generation, the latter child is destroyed immediately. This rule may result in having a population exceeding the population size with one individual. In

such situations, the population with the extra individual is advanced to the next generation. In the next generation, once the fitness values are calculated, the individual with the worst fitness value is discarded from the population. If the number of individuals with the worst fitness value is more than one, one of these individuals is chosen randomly.

The operation loading heuristic algorithm uses the same objective function of the original tool allocation model formulation to compute the fitness value corresponding to a chromosome.

4.3.1. Outline of the Heuristic Algorithm

The heuristic algorithm is developed to give a measure of the performance of a given tool-machine allocation.

Before the initiation of the algorithm, two types of information are present. The first one is the tool-machine allocations (a chromosome), that gives the types of tools in each machine's tool magazine. The second type of information is the list of operations that can be processed by each tool. This information is combined into the list of possible operation-machine allocations, which denotes which operation can be done on which machine(s).

The basic methodology of the proposed heuristic algorithm is choosing a part type and allocating its operations to alternative machines. Since the FMS considered in this study assumes the availability of alternative routes, alternative operations are possible for the part type to complete its flow in the system. Even though the methodology looks quite simple, there are a large number of alternative ways the decisions can be made while choosing these alternative operations and while allocating them on their alternative machines. The outline of the heuristic algorithm developed for operation loading given a tool allocation is presented in Table 4.12.

The decision steps of choosing a part type, choosing a route, choosing an operation and choosing a machine (steps (1), (2), (4) and (5)) are accepted to have the potential to be influential on heuristics performance. The alternative decision-making rules those can be applied on each decision step and their effects on the heuristic performance must be

analyzed in order to form up the optimum combination of decision-making rules of the algorithm to be used in fitness evaluation. Most of the alternative decisions are familiar and widely used in the literature.

Table 4.12. Pseudo code for operation loading heuristic algorithm

Available Part Types List *contains the list of part types, which are not already loaded on machines or not yet proven to be unloadable using the loading procedure.*
Route Availability *for a part type denotes whether there is at least one alternative route for a single part of a part type to be produced.*
Loading Size *for a part type is the number of parts that will be loaded on machines. It determines the workload that is assigned to a machine.*

(1) ***Choose a Part Type*** *from "Available Part Types List" if no part type can be chosen, STOP*
(2) ***Choose a Route*** *from the "Available Routes" of the part type*
(3) Set the remaining amount of parts of the chosen part type as the "Loading Size" for the operations in the route
(4) ***Choose an Operation*** *from the route*
(5) ***Choose a Machine*** *to load the chosen operation. If route's all operations are loaded go to (6), else, go to (4)*
(6) Update machines' available processing times and number of remaining parts, then go to (1)

4.3.2. Alternatives for Choosing a Part Type

While choosing a part type among the available part types list, the following five decision-making rule alternatives are considered.

- *Shortest Average Route Processing Time:* For each part type, the average of the alternative routes' processing times is multiplied by number of parts remaining. The smallest product is chosen.

- *Longest Average Route Processing Time:* For each part type, the average of the alternative routes' processing times is multiplied by number of parts remaining. The largest product is chosen.

- *Part Type Score, based on operations' number of alternative machines:* Part types' chances of completion for their available route's operations are increased by having many alternative machines, and decreased by having few alternative machines. The "Route Score" for a part type's route can be defined as the minimum of the number of alternative machines among all operations in the route. In other words, the route's operation with the least number of alternative machines, determines the route's score. For instance, Figure 4.1 displays the route score for R1 is 2, since R1's operation with the minimum number of alternative machines is Opr1A with 2 alternative machines. R2's route score corresponds to 1 when calculated in the same fashion.

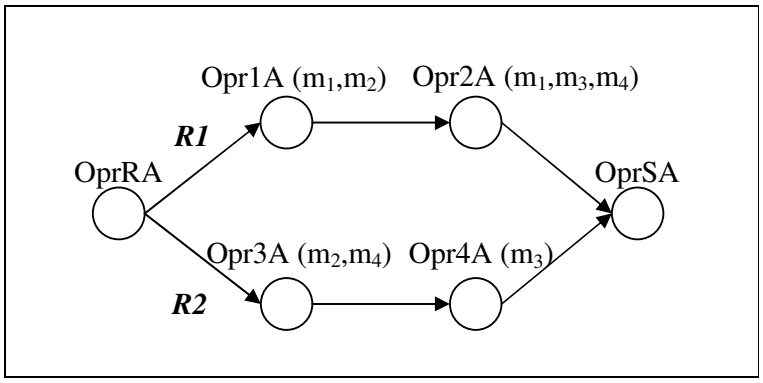


Figure 4.1. Demonstration of route score calculation

The minimum route score among all part type's is defined as the "Overall Minimum Route Score (OMRS)". Having defined these concepts, a part type's score (PTS) is calculated by the ratio:

$$(PTS)_i = \frac{(Number\ of\ Route\ Scores\ equal\ to\ OMRS)_i}{(Number\ of\ \ Available\ Alternative\ Routes)_i} \quad (4.12)$$

For instance, the PTS for A is equal to 0.5 (=1/2) if the OMRS is 1. The part type that has the maximum PTS is chosen for loading. This rule is developed so as to give loading priority to the part type that has less number of alternative machines for its operations.

- *Minimum Number of Alternative Routes:* The job with the minimum number of alternative routes is chosen.

- *Random Part Type Selection:* Part type to choose is determined randomly.

4.3.3. Alternatives for Choosing a Route

In the FMS design, it is assumed that a part can follow alternative operations in alternative routes, to come to its final condition. Once the part type is chosen, the question becomes, which route is to be followed. Three decision-making rule alternatives are considered at this stage.

- *Minimum Route Processing Time:* The processing time of a route is the total of the processing times of the operations of that route for a single part to be completed. The processing times of each route are calculated and the route with the smallest processing time is chosen.

- *Minimum Route Score:* Route scores are calculated as in *Part Type Score based on operations' number of alternative machines*. Among the available routes, the one with the minimum route score is chosen.

- *Random Route Selection:* Route alternative to choose is determined randomly.

4.3.4. Alternatives for Choosing an Operation

Each route is a sequence of operations. The order of the allocation of these operations may affect the quality of the allocation. The workloads assigned by different operations may be different and large workloads may prevent certain machines from further processing. Three decision-making rule alternatives are considered at this stage.

- *Original Route Sequence:* The allocation order of the route's operations is the same as the route sequence defined by the problem definition. The next operation from the sequence is chosen.

- *Operation with Minimum Number of Alternative Machines:* Among all the operations of the route, the one with the least number of alternative machines to be processed on is determined and chosen.

- *Random Operation Selection:* Operation alternative to choose is determined randomly among the remaining unassigned operations.

4.3.5. Alternatives for Choosing a Machine

Once the part type, the route, and the operation is determined, it is time for making the allocation on a machine. The machine must have enough available processing time to make the allocation. If there is only one available machine that has the tool which does the operation, the choice is straightforward. However, usually an operation can be done in several alternative machines having the required tool loaded on their tool magazines. A choice among these available alternative machines must be made in such occasions. Three decision-making alternatives are considered at this stage.

- *Machine with Maximum Available Processing Time:* Among all available possible machines, the one with the largest available processing time is chosen and operation is assigned on that machine.

- *Machine with Minimum Number of Alternative Operations:* The tools loaded on the machine's tool magazine indicate the operations that can be processed by the machine. In order to increase the utilization of dedicated machines in the system, for each machine, the operation alternatives other than the chosen operation are calculated. The machine which is capable of processing the fewest number of different operations is chosen. As a result, it is expected that the machines with greater capability of operation processing will not be occupied, and flexibility will be increased.

- *Random Machine Selection:* The machine to be assigned the chosen operation is determined randomly among the available alternative machines.

4.3.6. Choosing among Decision Alternatives

The best decision-making rule in each step of the heuristic algorithm is to be chosen from the aforementioned candidate decision alternatives. The selected rules will build up the structure of the proposed heuristic algorithm that will be used during fitness evaluation of solutions in the GA.

The heuristic giving the least (that is the best) fitness value is the one that performs closest to the optimal solution. In order to choose the best combination of rules that gives the best fitness value, in-dept analysis of the performance of each decision alternative is necessary. The summary of decision steps (factors) and their alternatives (factor levels) is given in Table 4.13. Ties are broken arbitrarily in every decision step of the algorithm.

A systematic approach is implemented for analyzing alternatives' performances. The experimentation process has been carried out for a number of FMS problem layouts and the factor levels that perform significantly better than the others are incorporated in the heuristic algorithm. The generation procedure for the test problem layouts is explained in Section 5.1 Test problem layouts can be observed in APPENDIX A.

For every experimental problem layout generated, a set of alternative feasible solutions is generated randomly and variants of the heuristic algorithm are applied on these solutions. The resulting fitness values make up the input data for the analysis.

Table 4.13. Heuristic algorithm's decision steps and alternatives

Factor Levels	Decision Step: Choose Part Type (A)
A1	Shortest Average Route Processing Time
A2	Longest Average Route Processing Time
A3	Part Type Score, based on operations' number of alternative machines
A4	Minimum Number of Alternative Routes
A5	Random Part Type Selection
Factor Levels	Decision Step: Choose Route (B)
B1	Minimum Route Processing Time
B2	Minimum Route Score
B3	Random Route Selection
Factor Levels	Decision Step: Choose Operation (C)
C1	Original Route Sequence
C2	Operation with Minimum Number of Alternative Machines
C3	Random Operation Selection
Factor Levels	Decision Step: Choose Machine (D)
D1	Machine with Maximum Available Processing Time
D2	Machine with Minimum Number of Alternative Operations
D3	Random Machine Selection

The analysis procedure can be summarized as determining the most influential decision steps of the algorithm and then choosing the best performing decision-making alternative for each decision step. This procedure is to be applied repeatedly until there is no influential decision step with its decision-making alternative remains unset.

4.3.7. Analysis Procedure for Determining the Heuristic Algorithm

The number of alternative combinations of decision-making alternatives is 135 (=5×3×3×3), which is equal to the multiplication of the number of decision-making rule alternatives for each decision step. Besides, for every problem layout, 100 feasible solutions, all of which are different from each other is generated and the fitness values of

all these solutions (chromosomes) corresponding to 135 rule combinations are calculated. Table 4.14 represents the resulting data set for each problem layout.

Table 4.14. Representation of the data set for each problem layout

Combination ID	Solution 1	Solution 2	…	Solution 100	Fitness Value Averaged Over All Solutions
A1-B1-C1-D1	$Y_{1,1}$	$Y_{1,2}$	…	$Y_{1,100}$	$\overline{Y}_{1j}$
A1-B1-C1-D2	$Y_{2,1}$	$Y_{2,2}$	…	$Y_{2,100}$	$\overline{Y}_{2j}$
A1-B1-C1-D3	$Y_{3,1}$	$Y_{3,2}$	…	$Y_{3,100}$	$\overline{Y}_{3j}$
A1-B1-C2-D1	$Y_{4,1}$	$Y_{4,2}$	…	$Y_{4,100}$	$\overline{Y}_{4j}$
…	…	…	…	…	…
A5-B3-C3-D3	$Y_{135,1}$	$Y_{135,2}$	…	$Y_{135,100}$	$\overline{Y}_{135j}$

The common tool used in finding the factors that have significant contribution to the output of a certain experiment is known as the “Analysis of Variance (ANOVA)”. By performing the ANOVA, the decision steps that cause the greatest variation in heuristic performance will be determined. Besides, the factor levels with best average response are denoted. All decision-making alternatives are treated as factor levels for this purpose.

The second step of the procedure, choosing the best decision-making alternative, is done getting use paired t-tests. The t-test is used to test whether two populations come from the same distribution. On the other hand, when the observations of two populations are pair wise dependent, paired t-test is used for checking whether the two populations come from the same distribution. The pair wise dependency of two values, in this respect will be the fact that they both are the averages of heuristic results of a number of combinations for a specific solution.

Data set organized for the paired t-test analysis for a factor’s levels (e.g. B1), is presented in Table 4.15. Let S_{FL} denote the set of combinations in the data set that has factor F set to level L (i.e. S_{B1} denotes the set of combinations in the data set that has factor B set to level 1). Each $\overline{Y}_{FL,j}$ corresponds to the average of fitness values for solution j

obtained as results of heuristic combinations S_{FL}. The row corresponding to the best factor level in Table 4.15 is tested pair wise with all other rows (factor levels) to see whether it has performed significantly better than its counterparts.

Table 4.15. Representation of the data set to be used in paired t-tests

Combination ID	Solution 1	Solution 2	…	Solution 100
S_{B1}	$\overline{Y}_{B1,1}$	$\overline{Y}_{B1,2}$	…	$\overline{Y}_{B1,100}$
S_{B2}	$\overline{Y}_{B2,1}$	$\overline{Y}_{B2,2}$	…	$\overline{Y}_{B2,100}$
S_{B3}	$\overline{Y}_{B3,1}$	$\overline{Y}_{B3,2}$	…	$\overline{Y}_{B3,100}$

The whole analysis procedure can be summarized as follows:

1. ANOVA is applied for the data set to determine the most influential factor and its factor level with the best average response.
2. Paired t-tests are applied to check whether the factor level chosen at 1 is significantly different (better) than the other factor levels.
3. If the factor level turns out to be significantly better performing than the others, its chosen level setting is fixed and the data set is updated by excluding the combinations other than this setting.

The above procedure is repeated until ANOVA does not indicate a significantly influential factor or t-test does not discriminate the factor levels in terms of their responses. For instance, 100 feasible solutions are generated for problem layout #1 and the factor level combinations are tested on these solutions. When ANOVA is applied using the initial data set, the Table 4.8 and Table 4.9 show that the decision step B (Choose Route) turns out to be the most influential factor with its best (minimum) solution at level B1 (i.e. Minimum Route Processing Time). If two-tailed paired t-test shows that B1 is significantly better than all other levels, B1 combination is set and all data including B setting different then B1 is excluded. The remaining data set is analyzed using ANOVA and the procedure is repeated in the same fashion.

Table 4.17 shows that B1 has the lowest (best) average fitness value among the others. So paired t-test is applied to check whether it is significantly better than the other

factor levels B2 and B3 over all generated solutions. Two t-tests are done in this respect. The first test checks if B1 and B2 have the same effect and the second test checks if B1 and B3 have the same effect. The hypothesis testing for t-tests is as follows:

H_0: B1's factor effect is not significantly different from B2 factor effect

H_1: Otherwise

p-value of the paired t-test is 0.00, so H_0 is rejected at 95% significance level.

H_0: B1's factor effect is not significantly different from B3 factor effect

H_1: Otherwise

p-value of the paired t-test: 0.00, so H_0 is rejected at 95% significance level.

$\Rightarrow$ It is concluded that B1 is significantly better performing than B2 and B3.

Table 4.16. 1st stage ANOVA table determining significance of factors (problem layout #1)

Source	SS	DF	MSE	F
SS_A	156.16	4	39.04	204.51
SS_B	268.18	2	134.09	702.42
SS_C	0.24	2	0.12	0.64
SS_D	0.53	2	0.26	1.39
SS_{error}	23.67	124	0.19	
SS_t	448.79			

Table 4.17. 1st stage average fitness values of factor levels (problem layout #1)

	Average Fitness Value
B1	9.14
B2	12.15
B3	12.10

The p-values for pair wise paired t-tests that compare the best performing factor level with the rest of the factor levels can also be summarized as in Table 4.18. Table 4.18 shows that B1's performance is significantly different than the other two levels since the p-

values for the t-tests for pairs (B1, B2) and (B1, B3) are less than 0.025 (significance level of 95%). So, B1 is fixed and the data set is updated such that combinations having B2 and B3 are excluded and 1st stage is concluded. ANOVA and t-tests are done repeatedly for the remaining factors A, C and D in the same fashion, until ANOVA or t-test does not come up with a unique influential factor.

Table 4.18. 1st stage p-values for paired t-tests among factor levels (problem layout #1)

	B1
B2	0.00
B3	0.00

Table 4.19, Table 4.20 and Table 4.21 summarize the most influential factors at each stage (1st, 2nd and 3rd) and the factor levels that give the best fitness value significantly different from other levels for each problem layout. The details such as ANOVA tables, average fitness values of factor levels, paired t-test results for all stages of every problem are presented in APPENDIX B.

The analysis procedure has revealed that, A (Choosing a Part Type) and B (Choosing a Route) are the most influential factors for each problem with their factor levels set at A1 and B1 respectively. Although the 3rd stage does not result in absolute influential factors for all problems or their significantly better levels, factors D and C are set at levels C2 and D1, respectively. The reason for choosing C2 is that, among the three problems that C has turned out to be influential, all has C2 with better (less) fitness value, even though C2 is significantly different than other levels in two of these problems. The reason for setting D to level D1 is that, in two out of three problems, D1 results as the best factor level, although it is not significantly different than others in one of these two. The resulting decision-making rules concluded to be chosen for the heuristic algorithm decision steps are presented in Table 4.22.

Once the generic structure of the heuristic algorithm is determined, further chances of improvement on the algorithm are to be studied. First, tie breaking strategies for the decision steps are analyzed. Secondly, altering the allocation size of operations which is

originally set as assigning the remaining amount in the fitness evaluation heuristic algorithm is considered.

Table 4.19. 1st stage summary of analysis procedure for determining the heuristic algorithm

1st Stage			
Problem Layout	Significant Factor with Greatest Influence	Best Factor Level	t-test Result
P1	B	B1	Significantly different
P2	B	B1	Significantly different
P3	A	A1	Significantly different
P4	B	B1	Significantly different
P5	A	A1	Significantly different
P6	B	B1	Significantly different
P7	B	B1	Significantly different
P8	A	A1	Significantly different
P9	A	A1	Significantly different
P10	A	A1	Significantly different

Table 4.20. 2nd stage summary of analysis procedure for determining the heuristic algorithm

2nd Stage			
Problem Layout	Significant Factor with Greatest Influence	Best Factor Level	t-test Result
P1	A	A1, A4	Not different
P2	A	A1	Significantly different
P3	B	B1	Significantly different
P4	A	A1	Significantly different
P5	B	B1	Significantly different
P6	A	A1	Significantly different
P7	A	A1	Significantly different
P8	B	B1	Significantly different
P9	B	B1	Significantly different
P10	B	B1	Significantly different

Table 4.21. 3rd stage summary of analysis procedure for determining the heuristic algorithm

3rd Stage			
Problem Layout	Significant Factor with Greatest Influence	Best Factor Level	t-test Result
P1	-	-	-
P2	none	-	-
P3	none	-	-
P4	D	D2	Significantly different
P5	C	C2	Significantly different
P6	C, D	C2, D1	C2 is significantly different, D1 is not
P7	D	D1	Significantly different
P8	C	C2	Not different
P9	none	-	-
P10	none	-	-

Table 4.22. Decision-making rules chosen as a result of analysis procedure

Decision Step	Chosen Factor Level	Decision-Making Rule
Choose Part Type (A)	A1	Shortest Average Route Processing Times
Choose Route (B)	B1	Minimum Route Processing Time
Choose Operation (C)	C2	Operation with Minimum Number of Alternative Machines
Choose Machine (D)	D1	Machine with Maximum Available Processing Time

4.3.8. Employing Tie Breaking Strategies

Tie breaking strategies in every decision step of the algorithm has been set as making a random selection among the candidates during the experimentations done so far. However, a smarter tie breaking strategy would increase to the quality of results. The decision-making rules that are observed to be the second most influential are tested for this purpose. For this purpose, the second best factor levels for every stage done in section

4.3.6 are determined and are checked for significance. Each stage of analysis and the results are displayed in Table 4.23, Table 4.24 and Table 4.25.

The results of this additional procedure do not provide any information revealing the fact that any second best factor can be selected as a tie breaking strategy.

- B2 and B3 cannot be differentiated in six out of ten problems.
- Factor A levels are also indifferent in five out of ten problems. Moreover, the rest of the ten problems do not indicate a dominance of a second best factor level.
- Section 4.3.6's unset factor levels C1 and D2 cannot be differentiated from C3 and D3, which are random selections, respectively.

As a result, no additional tie breaking strategies are included in the heuristic algorithm and ties are broken arbitrarily at each decision step.

Table 4.23. 1st stage summary of tie breaking strategies analysis procedure for determining the heuristic algorithm

1st Stage		
Problem Layout	Second Best Factor Level	t-test Result
P1	B3, B2	Not different
P2	B3, B2	Not different
P3	A3, A4, A5	Not different
P4	B3	Significantly different
P5	A4	Significantly different
P6	B2	Significantly different
P7	B2	Significantly different
P8	A3	Significantly different
P9	A5, A3	Not different
P10	A3, A5	Not different

Table 4.24. 2nd stage summary of tie breaking strategies analysis procedure for determining the heuristic algorithm

2nd Stage		
Problem Layout	Second Best Factor Level	t-test Result
P1	-	-
P2	A5	Significantly different
P3	B2, B3	Not different
P4	A3, A5	Not different
P5	B3	Significantly different
P6	A3	Significantly different
P7	A3, A4	Not different
P8	B3, B2	Not different
P9	B3, B2	Not different
P10	B3, B2	Not different

Table 4.25. 3rd stage summary of tie breaking strategies analysis procedure for determining the heuristic algorithm

3rd Stage		
Problem Layout	Second Best Factor Level	t-test Result
P1	-	-
P2	-	-
P3	-	-
P4	D1, D3	Not different
P5	C3	Significantly different
P6	C3, C1	Not different
P7	D2, D3	Not different
P8	-	-
P9	-	-
P10	-	-

4.3.9. Altering the Allocation Size

The third step of the Generic Fitness Evaluation Algorithm involves the assignment of the remaining amount of the part type's chosen operation on the chosen machine. This step of the algorithm is modified to search for further improvement in the fitness values.

The number of parts to be flowed through the chosen route is an important parameter in many aspects. The size of allocation multiplied by the corresponding processing time determines the workload assigned to a machine. Larger sizes of allocations will occupy greater available processing times of machines and reduce their chances of processing other operations.

Smaller sizes of allocation would definitely be a remedy for this problem. However, too much reduction of the sizes will result in large computation times, due to the increase in number of times of allocations. Each allocation corresponds to an iteration in the heuristic procedure.

The below alternatives for determining the sizes of allocation are proposed and tested against the present allocation procedure that allocates all of the remaining parts. The allocation sizes in the below alternatives are chosen as a function of the remaining number of parts of the current part type in the system.

- *Random Allocation Size Selection (type I):* The allocation size is determined randomly such that it is between ½ and total of the remaining number of parts. The allocation size is determined using the below formula.

  $$A = \lceil 0.50 \times \text{Remaining Size} \rceil$$

  $$B = \text{Remaining Size}$$

  $$\text{Allocation Size} = A + \text{Random}(B - A)$$

  where Random(X) function returns random integer values from 1 to X.

- *Random Allocation Size Selection (type II):* The allocation size is determined randomly such that it is between 1 and total of the remaining number of parts, using the below formula.

  $$A = 1$$
  $$B = \text{Remaining Size}$$
  $$\text{Allocation Size} = A + \text{Random}(B - A)$$

- *Random Allocation Size Selection (type III):* The allocation size is determined randomly such that it is between ¼ and ¾ of the remaining number of parts, using the below formula.

  $$A = \lceil 0.25 \times \text{Remaining Size} \rceil$$
  $$B = \lceil 0.75 \times \text{Remaining Size} \rceil$$
  $$\text{Allocation Size} = A + \text{Random}(B - A)$$

- *Random Allocation Size Selection (type IV):* The allocation size is determined randomly such that it is between 1 and ½ of the remaining number of parts, using the below formula.

  $$A = 1$$
  $$B = \lceil 0.50 \times \text{Remaining Size} \rceil$$
  $$\text{Allocation Size} = A + \text{Random}(B - A)$$

- *One by One Allocation:* Each time, a single part is allocated.

  $$\text{Allocation Size} = 1$$

  Decimal points are rounded up to the next integer.

Experiments involving the aforementioned alternatives have been carried out to see the effect of reducing size on fitness value, as well as its effect on run time. The decision alternative of the allocation size is chosen to be factor E and the corresponding factor levels are displayed in Table 4.26.

Table 4.26. Decision alternatives (factors) and their levels for the allocation size

Factor Levels	Decision Step: Choose Allocation Size (E)
E1	Allocation of All of the Remaining Parts
E2	Random Allocation Size Selection (type I) (Random between Half to All of Remaining Parts)
E3	Random Allocation Size Selection (type IV) (Random between One to All of Remaining Parts)
E4	Random Allocation Size Selection (type II) (Random between Quarter to Three Quarters of the Remaining Parts)
E5	Random Allocation Size Selection (type III) (Random between One to Half of the Remaining Parts)
E6	One by One Allocation

For an alternative to be eligible for implementing in the heuristic, the below performance measures are considered:

- Having a tolerable run time
- Showing good performance

The mathematical model for operation loading presented at the beginning of this section is developed in GAMS and solved using the CPLEX 9.1 solver. Even it may take a large amount of time; CPLEX has the ability to compute the optimal solution for a loading problem. The heuristic algorithm developed in this study is supposed to come up with a result in a shorter period of time. So, in order to determine the best factor level for choosing the allocation size, each alternative's runtime as well as their average performance (fitness) are compared with each other and with those of the CPLEX algorithm.

So, in order to determine the best factor level for choosing the allocation size, each alternative's runtime as well as their average performance (fitness) are compared with each other and with those of the CPLEX algorithm. The model described in section 4.3 is solved using CPLEX and the heuristic with alternatives E1 to E6 for 100 randomly generated

solutions and the results are recorded (CPLEX runs are finalized if they have not reached an optimal value for 1000 sec).

Table 4.27. Average runtimes for CPLEX and heuristics with alternative allocation sizes for all problems

	Average Run Time (sec)									
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
CPLEX	0.57	0.56	0.58	0.61	46.19	1.31	38.70	0.95	1.17	0.85
E1	0.03	0.04	0.07	0.10	0.28	0.35	0.38	0.68	0.97	0.77
E2	0.04	0.05	0.10	0.17	0.39	0.46	0.55	0.96	1.48	1.18
E3	0.05	0.06	0.12	0.20	0.47	0.55	0.66	1.15	1.76	1.40
E4	0.06	0.07	0.12	0.22	0.48	0.57	0.70	1.23	1.99	1.52
E5	0.08	0.09	0.18	0.32	0.66	0.81	0.98	1.70	2.70	2.17
E6	0.14	0.17	0.30	0.60	1.05	1.38	1.45	2.50	4.02	3.55

Table 4.28. Average fitness values for CPLEX and heuristics with alternative allocation sizes for all problems

	Average Fitness Values									
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
CPLEX	3.51	9.92	6.44	9.93	5.26	8.17	4.75	20.84	9.70	7.83
E1	8.09	16.42	14.25	19.39	12.18	15.42	13.96	39.24	29.53	26.77
E2	7.07	15.40	12.52	17.85	11.08	14.23	12.57	36.68	27.08	23.35
E3	6.65	14.68	11.49	17.08	10.73	13.59	11.92	35.44	26.24	21.55
E4	6.34	14.50	11.30	16.70	10.45	13.13	11.65	34.66	25.92	20.51
E5	6.13	14.21	11.20	16.36	10.29	12.83	11.47	34.21	25.56	19.95
E6	6.00	13.93	11.04	16.18	10.20	12.55	11.35	33.71	25.16	19.31

Table 4.27 and Table 4.28 also prove that even though the exact solution algorithm has better fitness values than the heuristic alternatives; the computation time remains prohibitive in some cases (like in problem 5 and 7). Besides, there is no guarantee of an exact algorithm for finding a feasible solution in a reasonable time. It is very common that the CPLEX may not able to find a feasible solution to a problem after a computation period of 24 hours.

It is observed that the runtime of E1 is always lower than that of the CPLEX algorithm. However, its average performance is the poorest one among all alternatives. For this reason, the factor with the next minimum computation time, E2 is decided to be considered as a candidate to be applied in the operation loading algorithm. At this stage, the extra computation time resulted by employing E2 is accepted to be tolerable.

To see if E2 performs significantly better than E1 in terms of average fitness values, 100 randomly generated solutions for each of ten problems are evaluated using the two algorithms, one employing E1, the other E2. Paired t-tests are applied on resulting fitness values to check whether one performs significantly better than the other. The paired t-test results are displayed in the below table:

Table 4.29. Paired t-test results for E1 and E2

	Average Fitness Value			
Problem	E1	E2	p-values	t-test Result
P1	8.089	7.189	0.000	Significantly different
P2	16.421	15.512	0.001	Significantly different
P3	14.245	12.746	0.000	Significantly different
P4	19.391	18.276	0.000	Significantly different
P5	12.178	11.595	0.000	Significantly different
P6	15.419	14.836	0.000	Significantly different
P7	13.956	13.706	0.000	Significantly different
P8	39.240	37.197	0.000	Significantly different
P9	29.532	27.524	0.000	Significantly different
P10	26.774	24.887	0.000	Significantly different

Table 4.29 shows that in all ten problems E2 gives significantly different and better (lower) results than E1. As a result, E2 is selected as the strategy to determine the allocation size.

4.3.10. The Finalized Heuristic Algorithm

The resulting decision making rules for decision steps of the developed heuristic algorithm are shown in Table 4.30. The finalized heuristic algorithm for determining the fitness value of a tool configuration is presented in Table 4.31.

The FMS layout information gives the initial number of parts and machine availabilities in the system before the heuristic algorithm is executed. In order to observe the overall performance of the finalized heuristic algorithm for the loading problem, the results obtained by running the heuristic algorithm are compared with optimal results of the loading problem solved by CPLEX. For this purpose both heuristic algorithm and CPLEX are executed for 100 randomly generated feasible solutions and the fitness values are analyzed for ten problems. The analysis procedure compares the final fitness value reached by executing the loading alternative (the heuristic and the CPLEX) with the initial fitness value, without executing any loading alternative.

Table 4.30. Decision-making rules chosen for the finalized heuristic algorithm

Decision Step	Chosen Factor Level	Decision-Making Rule
Choose Part Type (A)	A1	Shortest Average Route Processing Times
Choose Route (B)	B1	Minimum Route Processing Time
Choose Operation (C)	C2	Operation with Minimum Number of Alternative Machines
Choose Machine (D)	D1	Machine with Maximum Available Processing Time
Choose Allocation Size(D)	E2	Random Allocation Size Selection (type I) (Random between Half to All of Remaining Parts)

Table 4.31. Pseudo code for the finalized heuristic algorithm

Available Part Types List *contains the list of part types, which are not already loaded on machines or not yet proven to be unloadable using the loading procedure.*

Route Availability *for a part type denotes whether there is at least one alternative route for a single part of a part type to be produced.*

Loading Size *for a part type is the number of parts that will be loaded on machines. It determines the workload that is assigned to a machine.*

(1) ***Choose the Part Type*** *with Shortest Average Route Processing Times from Available Part Types List. If no part type can be chosen, STOP*

(1.1) Check routes' availabilities for all part types in the Available Part Types List

(1.2) If no route is available for a part type, discard it from the Available Part Types List, else ***Choose a Part Type*** *(using Shortest Average Route Processing Times) from the Available Part Types List*

(1.3) If a part type can be chosen, go to (2) else ***STOP***

(2) Choose the Route with Minimum Route Processing Time from the available routes of the part type

(2.1) If no route is available for the part type, discard the part type from the Available Part Types List and go to (1)

(2.2) If at least one route is available for the part type, ***Choose the Route*** *with Minimum Route Processing Time from the available routes of that part type and go to (3)*

(3) Set the Loading Size for the operations in the route to a Random Size between Half to All of Remaining Parts

Table 4.31 continued

(4) ***Choose the Operation*** *with Minimum Number of Alternative Machines from the route*

(4.1) ***Choose the Operation*** *with Minimum Number of Alternative Machines from the route which is not already loaded on any machine and go to (5)*

(5) ***Choose the Machine*** *with Maximum Available Processing Time to load the chosen operation. If route's all operations are loaded go to (6), else, go to (4)*

(5.1) ***Choose the Machine*** *with Maximum Available Processing Time, such that it has the tool required to make the operation and has available processing time to complete the chosen operation of at least one part, among the available machines*

(5.2) Determine the maximum number of operations those can be loaded on the machine

(5.3) If the maximum number of loadable operations is less than the loading size, set loading size equal to maximum number of loadable operations

(5.3) If all operations of the route have been loaded on a machine go to (5.4), else and go to (4)

(5.4) If it turns out parts all operations can be loaded, load on their chosen machines, else it means that the route is no longer available

(5.5) Go to (6)

(6) Update machines' available processing times and number of remaining parts, then go to (1)

(6.1) If any loading has been done update machines' available processing times with respect to the number of operations loaded on machines and number of remaining parts for the part type. If all parts are completed, discard it from the Available Part Types List

(6.3) Go to (1)

Worst possible fitness value for a problem can easily be calculated by putting initial quantities of part types in the problem using the number of uncompleted parts (S_i) in the objective function. For instance if we consider problem 1 (P1), in order to calculate the initial fitness value, the maximum initial quantity ($q_B = 11$) is multiplied by the completion ratios of all part types and increased by average of the pair wise differences of all machines' utilizations.

$$WFV_{P1} = \max_i\{q_i\} \times \left[\sum_i \left(\frac{S_i}{q_i} \right) \right] + \left[\frac{\sum_{\forall (m,m') \in PAIRS(m)} \left(U^+_{mm'} + U^-_{mm'} \right)}{\frac{Card(m) \times [Card(m) - 1]}{2}} \right] \tag{4.13}$$

$$WFV_{P1} = 11 \times \left[\frac{10}{10} + \frac{11}{11} + \frac{7}{7} \right] + \left[\frac{\frac{130-130}{130} + \frac{130-130}{130} + \frac{130-130}{130}}{\frac{3 \times 2}{2}} \right] \tag{4.14}$$

$$WFV_{P1} = 11 \times [3] + [0] = 33 \tag{4.15}$$

The operation of the heuristic algorithm for a problem can be viewed as making incremental improvements over the worst fitness values. The heuristic algorithm allocates part types' operations on machines (loads the machines) and the fitness value is improved (decreased) with every allocation. The performance of a loading algorithm (heuristic, CPLEX, etc.) can be interpreted as the degree of improvement over the initial fitness value.

Degree of improvement can be calculated by the formula:

$$\left(Per\ Cent\ Improvement \right)_{Algorithm} = \frac{WFV - \left(Fitness\ Value \right)_{Algorithm}}{WFV} \tag{4.16}$$

As the fitness value is decreased, the above ratio increases and improvement is accomplished.

For these reasons, Average Fitness Values (AFV) for the Heuristic Algorithm (HA) and CPLEX, together with their per cent improvements are calculated.

Table 4.32. Average fitness values and per cent improvements in average fitness values by the developed heuristic algorithm and the CPLEX

Problem	AFV of HA	AFV of CPLEX	Worst Fitness Value	% Improvement in AFV by the HA	% Improvement in AFV by the CPLEX	Relative Performance
P1	7.07	3.51	33	78.58%	89.36%	87.93%
P2	15.4	9.92	51	69.80%	80.55%	86.66%
P3	12.52	6.44	48	73.92%	86.58%	85.37%
P4	17.85	9.93	60	70.25%	83.45%	84.19%
P5	11.08	5.26	70	84.17%	92.48%	91.01%
P6	14.23	8.17	75	81.03%	89.10%	90.94%
P7	12.57	4.75	70	82.04%	93.22%	88.01%
P8	36.68	20.84	140	73.80%	85.12%	86.70%
P9	27.08	9.70	140	80.66%	93.07%	86.66%
P10	23.35	7.83	120	80.54%	93.47%	86.17%
					Average	87.36%

The last column of Table 4.32 shows the ratio of per cent improvements of HA and CPLEX. The average relative performance in terms of fitness values of 87.36% and tolerable runtimes are considered to be satisfactory for the developed heuristic algorithm to be used as the fitness evaluation procedure for the GA implementation.

Figure 4.2 is a representation of the performance of the heuristic algorithm. 100 randomly generated chromosomes' fitness values are computed by loading the operations by both the developed heuristic algorithm and CPLEX. The resulting values are ordered with respect to the CPLEX fitness values.

Figure 4.2 shows that the heuristic works as a good surrogate for the model to evaluate fitness of a given tool allocation.

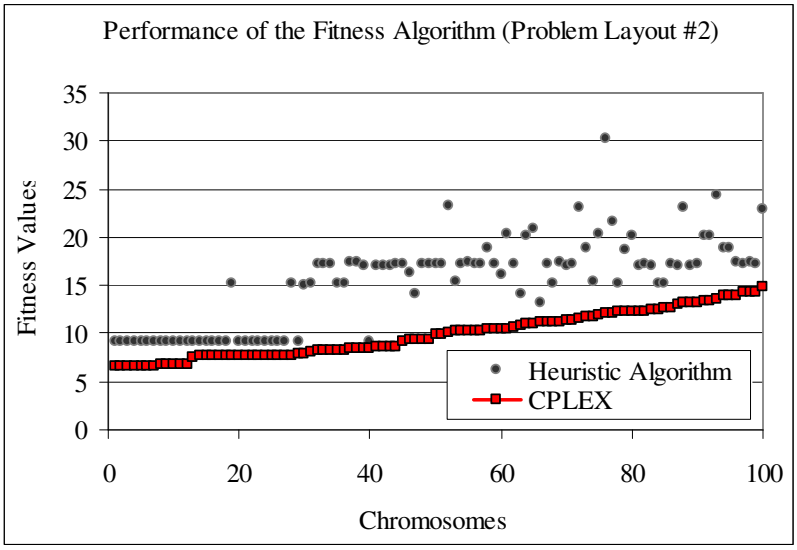


Figure 4.2. Comparison of heuristic and CPLEX results for problem layout #2

4.4. Selection

Selection defines the parents whose chromosomes will be passed to next generation. Survival-of-the-Fittest asserts that the "fit" individuals are the ones that are more likely to survive and reproduce. For this reason, the GA approach to the tool allocation problem uses a selection procedure related to fitness values of the solutions in the population like most GA applications do. Applying the elitist strategy, the individual with the best fitness value is preserved for the next population. If there is more than one solution, different from each other with the same fitness value, one of them is chosen arbitrarily. The selection procedure of the GA is depicted in Table 4.33.

Table 4.33. Pseudo code for the selection in proposed GA

…

(4) Preserve the best solution in the population and calculate the selection probabilities

(5) While number of children produced is less than population size

(5.1) ***Select two mates*** *from the population*

…

A solution is not allowed to mate with itself. The selected solutions, namely the parents, are the ones that probably have advantageous genetic structure and their chromosomes will be passed at least to the next generation. Parents' chromosomes are passed either partially or entirely to the next generation due to the probabilistic nature of genetic operations that they are exposed to.

Different schemes are applicable for selection of parents in GA like "Roulette Wheel Selection (RWS)" that chooses a single individual at a time, with a probability determined by its fitness value. The procedure resembles spinning a roulette wheel where the wheel is partitioned into slices proportional to the fitness values of individuals. Moving the spinner by a randomly generated angle between zero and 2π, from a reference point, makes the selection of an individual. Another scheme for selection is the "Stochastic Universal Selection (SUS)", which selects all parents simultaneously by using a spinner that has a number of arms. The number of mates selected by SUS is equal to the number of equally spaced arms on the spinner.

RWS is chosen to be the selection method for this study. The number of selections to be made at each generation is not definite due to the "no duplicates" nature of the population. If the child that is formed as a result of crossover-and-mutation strategy is a duplicate of another child which is already formed, only one of these twins is preserved and the other is destroyed. For this reason, as long as duplicates are generated within the procedure, mates are chosen to fill the next population using RWS.

The selection probabilities that are used in the RWS of individuals in the population can be calculated by using a number of alternative methods. Scaling method makes a linear transformation of the fitness values. Ranking method ranks the individuals with respect to their fitness values and scales these ranks. Tournament selection method involves choosing a set of individuals, comparing them with each other and selecting the best one as a mate. Among these alternatives, scaling and ranking methods are applied in the GA.

The two alternative selection probability calculation procedures considered for selection procedure are:

- Fitness-Based Selection Probability Calculation (Scaling Method)
- Ranking-Based Selection Probability Calculation (Ranking Method)

4.4.1. Fitness-Based Selection Probability Calculation

Each individual's selection probability is based on its fitness value calculated by the heuristic algorithm. Since the objective is minimization, in order to assign greater probabilities to individuals with less fitness values, probabilities are assigned with respect to the differences of individuals' fitness vales from maximum (worst) fitness in the population. This procedure by itself is expected to intensify the population into the individuals very similar to the ones with largest fitness values.

Figure 4.3 and Table 4.34 presents an illustration of the fitness-based selection probability calculation. Consider a population of 10 individuals. White bars in the figure correspond to the fitness values of these individuals. 7^{th} individual has the worst (maximum) fitness value. Dark bars, which represent the differences between the maximum fitness value and the individuals' fitness values is used to compute the probabilities. The selection probability of an individual is equal to the ratio of its dark bar's length over total length of dark bars over the whole population. It is obvious that the 5^{th} individual which has the lowest (best) fitness value will have the greatest selection probability.

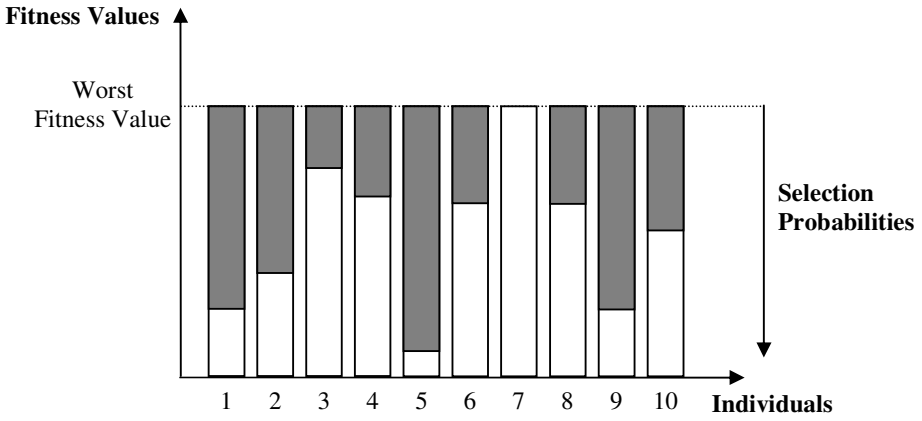


Figure 4.3. Illustration of the fitness-based selection probability calculation

Table 4.34. Pseudo code for fitness-based selection probability calculation

```
MFV : The maximum (worst) fitness value
T   : MFV multiplied by population size
S   : Sum of all fitness values in the population
FV_i : Fitness value of i^th individual
R   : A random number generated between 0 and (T- S)

    current_sum = 0
    i = 1
    while ( true )
    {   current_sum = current_sum + ( MFV - FV_i )
        if ( current_sum > R ), return i
        else, i = i +1
    }
```

4.4.2. Ranking-Based Selection Probability Calculation

Ranking procedure almost eliminates the effect of fitness values in the selection. When the individuals are ranked with respect to their fitness values, the amount of difference in fitness values is not considered, which in turn results in a more diversified population. The differences in selection probabilities for ranking–based calculation are much less than those of the fitness-based calculation most of the time.

Where N denotes the size of the population, the probability of selecting an individual can be assumed to be $P(k) = \alpha + \beta k$ where α and β are scalars.

For P(k), to be a probability distribution, it is required to satisfy:

$$\sum_{k=1}^{N} (\alpha + \beta k) = 1 \;\Rightarrow\; N\left(\alpha + \beta \frac{N+1}{2}\right) = 1 \tag{4.17}$$

For ranking-based selection, the "Selection Pressure" can be defined as:

$$\phi = \frac{\alpha + \beta N}{\alpha + \beta\left(\frac{N+1}{2}\right)} \tag{4.18}$$

which is equivalent to the ratio: $\frac{P\left(Selecting\ fittest\ individual\right)}{P\left(Selecting\ median\ individual\right)}$ (4.19)

Some linear algebra established the following equations for α and β.

$$\beta = \frac{2\left(\phi - 1\right)}{N\left(N - 1\right)} \text{ and } \alpha = \frac{2N - \phi\left(N+1\right)}{N\left(N - 1\right)} \text{ implying that } 1 \le \phi \le 2 \tag{4.20}$$

The cumulative probability distribution defined using the above can be stated as:

$$P(k) = \alpha k + \beta\frac{k(k+1)}{2} \tag{4.21}$$

Having generated a random value R, the above equation can be equated to R resulting in the below equation, that determines the ranking of the individual to be selected for reproduction.

$$k = \frac{-(2\alpha + \beta) + \sqrt{(2\alpha + \beta)^2 + 8\beta r}}{2\beta} \tag{4.22}$$

Selection probability ϕ set to 1 means that all ranking effect is not considered and individuals' selection probabilities are equal to each other ($P(k) = 1/N$).

The selection pressure for the tool allocation problem is chosen to be 2, so that individuals with high ranks are given more chances to reproduce.

Table 4.35. Pseudo code for ranking-based selection probability calculation

Ranking-Based Selection Procedure

R : A random number generated between 0 and 1

Rank all individuals with respect to their fitness values

Return $k = \dfrac{-(2\alpha+\beta)+\sqrt{(2\alpha+\beta)^2+8\beta r}}{2\beta}$

These two alternatives for selection probability calculation will be tested for their performances.

4.5. Crossover

The crossover strategy applied in this study is the crossover-and-mutation, which means that the mates that are selected for reproduction first go through crossover and then mutation with relevant probabilities of crossover and mutation. Crossover makes a recombination of the genes of two mates. It is simply an exchange of genes (or alleles) between two mates. The crossover procedure for the developed GA is as follows:

Table 4.36. Pseudo code for crossover in proposed GA

…

(5.2) ***Apply crossover*** *on two mates to form two children*

(5.2.1) If crossover condition is satisfied

(5.2.1.1) Crossover operation is applied and the two selected mates become two children

(5.2.1.2) If crossover has caused any violation in feasibility conditions, a repairing procedure is applied for each child

…

The crossover condition implies the random nature of crossover operation. If a randomly generated number between 0 and 1 is less than the “Crossover Probability”, the crossover condition is satisfied. Crossover probability is a problem-specific value and set by analyzing the experimental results. Depending on the crossover probability, if crossover is applied on the mate, the child becomes genetically recombined. However, if crossover is not applied on the mate, the child becomes a replica of the mate.

Although various types of crossover operators have been defined in the literature, none of these operators seem to be appropriate for the matrix-based representation with feasibility constraints developed in this study.

Two alternative types of crossover procedures are developed for the matrix-based chromosomes. These procedures will be tested for their performances:

- Tool Configuration (Row) Exchange Crossover
- Machine Configuration (Column) Exchange Crossover

4.5.1. Tool Configuration (Row) Exchange Crossover

Tool configuration exchange crossover is simply done by choosing a row index of from the chromosome randomly and exchanging the rows with that index of the mates’ chromosomes.

The resulting solution of crossover procedure may require “Repairing” due to the infeasibility which may have occurred due to the exchange. Since initial distributions of tool copies are feasible before any exchange is done, it is obvious that doing the tool exchange does not cause infeasibility in tool availability constraints. However, feasibility checks for columns (i.e. slot capacities constraints) are necessary. If the slot capacity for a machine is exceeded for a solution after the crossover, the tools on that machine must be reorganized to maintain feasibility. Reorganizing (i.e. repairing) of a machine’s slots is done simply by choosing a random tool from set of loaded tools on that machine, other than the tool whose tool configurations have been exchanged, and unloading that tool from the machine.

Table 4.37 summarizes whether a repairing procedure is required after the crossover of tool type (or row) i. The values of *(i,j)*'s denote the binary numbers in cells on the row where the exchange takes place.

Table 4.37. Repairing conditions for tool exchange crossover

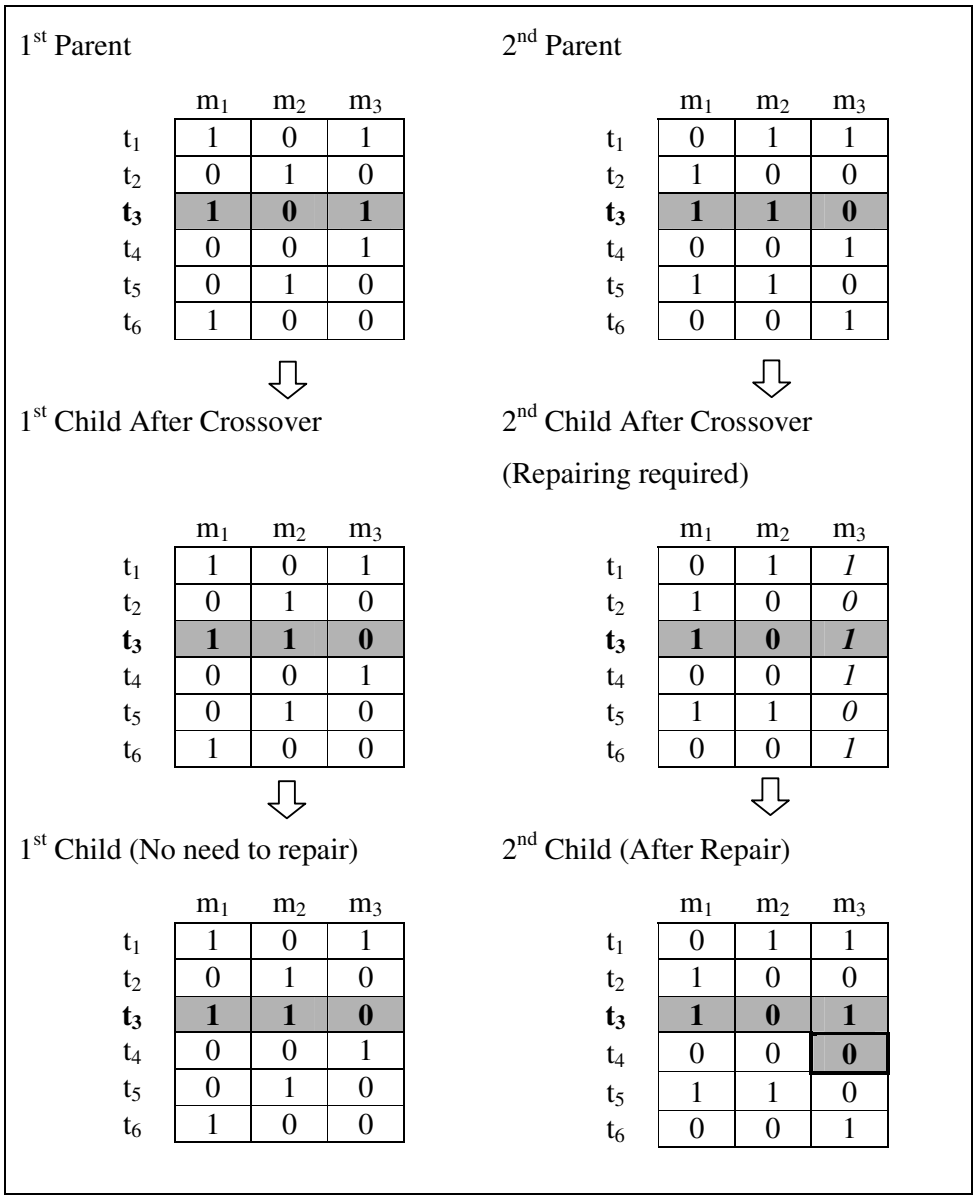
Original Value of *(i,j)*	Value of *(i,j)* after Crossover	Action
1	0	No Repairing Required
1	1	No Repairing Required
0	1	Repairing may be required

Table 4.38 is an illustration of the tool configuration exchange crossover procedure, where slot capacities of machines and tool availabilities are set to 3 and 2, respectively. Tool sizes are equal to 1.

Note that, 1st Child (After Crossover) does not require a repair even though the cell *(3,2)*, originally set to 0, has become 1 after crossover. This is due to the fact that slot capacity constraint of m_2 has not originally been a tight constraint (i.e. there has been an available slot in the magazine). Thus, loading t_3 on m_2 has not caused infeasibility after the crossover operation.

On the other hand, the situation is not same with 2nd Child (After Crossover), where the slot capacity constraint for m_3 is violated after the crossover. So a randomly selected tool from set of loaded tools on m_3 except for t_3 (i.e. t_1, t_4, t_6), is unloaded from m_3 (e.g. t_4 is unloaded in the illustration).

Table 4.38. Illustration of tool configuration exchange crossover



	m_1	m_2	m_3
t_1	1	0	1
t_2	0	1	0
t_3	**1**	**0**	**1**
t_4	0	0	1
t_5	0	1	0
t_6	1	0	0

	m_1	m_2	m_3
t_1	0	1	1
t_2	1	0	0
t_3	**1**	**1**	**0**
t_4	0	0	1
t_5	1	1	0
t_6	0	0	1

	m_1	m_2	m_3
t_1	1	0	1
t_2	0	1	0
t_3	**1**	**1**	**0**
t_4	0	0	1
t_5	0	1	0
t_6	1	0	0

	m_1	m_2	m_3
t_1	0	1	*1*
t_2	1	0	*0*
t_3	**1**	**0**	***1***
t_4	0	0	*1*
t_5	1	1	*0*
t_6	0	0	*1*

	m_1	m_2	m_3
t_1	1	0	1
t_2	0	1	0
t_3	**1**	**1**	**0**
t_4	0	0	1
t_5	0	1	0
t_6	1	0	0

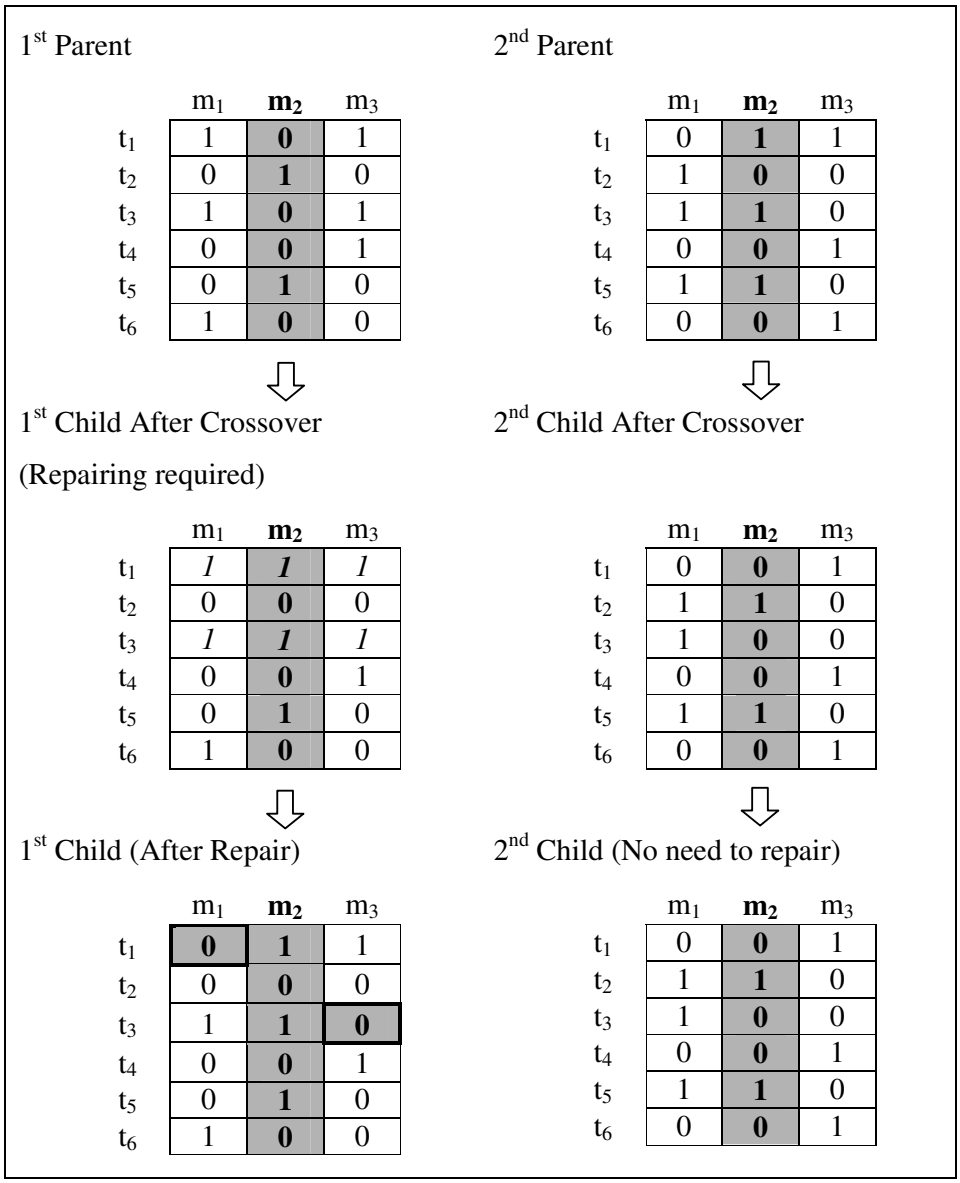
	m_1	m_2	m_3
t_1	0	1	1
t_2	1	0	0
t_3	**1**	**0**	**1**
t_4	0	0	**0**
t_5	1	1	0
t_6	0	0	1

4.5.2. Machine Configuration (Column) Exchange Crossover

Machine configuration exchange crossover is very similar to the tool configuration exchange crossover. The operation is exchanging the tool magazine configurations of a specific machine among two mates. For this purpose, a machine (a column of the chromosome) is randomly chosen and its tool magazine configurations in two mates are switched. Even though the slot capacity constraints are not violated, this procedure may affect the tool availability constraints. The resulting child may have an extra tool loaded

on its machines which does not actually exist. Similar to the tool exchange case, a repairing procedure may be required to resolve this situation. Repairing is done by first determining the tools whose tool availability constraints are violated and then unallocating those tools from machines which are chosen arbitrarily. While making the unallocation, the randomly chosen machine should be other than the machine whose tool magazine configurations are exchanged. Table 4.39 illustrates the machine configuration exchange crossover and its repairing procedure. Slot capacities of machines and tool availabilities are set to 3 and 2, respectively. Tool sizes are 1.

Table 4.39. Illustration of machine configuration exchange crossover



2^{nd} Child does not require repairing procedure even though the cell (2,2), originally set to 0, has become 1 after crossover, since the tool availability constraint of t_2 has not originally been a binding constraint (i.e. a tool copy of t_2 has been idle). Thus, loading t_2 on m_2 has not caused infeasibility after the crossover operation.

On the other hand, the situation is not same with the 1^{st} Child. After the crossover procedure, the tool availability constraints of both t_1 and t_3 become violated. For t_1, among m_1 and m_3, one of them is chosen arbitrarily (m_1, in the illustration) and unallocated from that machine. For t_3, the repairing procedure is repeated.

4.6. Mutation

The children produced by mates that have gone through crossover with crossover probability are exposed to mutation. Mutation is applied on each child independently and probabilistically with a mutation probability in order to produce genetic diversity in the chromosomes. The best level of mutation probability is problem-specific and will be set as a result of experimentation process.

Mutation makes perturbations in the alleles of the chromosome it is applied to. Mutation procedure developed for the matrix-based chromosome of the tool allocation problem is simply altering the value of an allele that corresponds to a randomly chosen tool-machine combination. In other words, a cell from the matrix is randomly chosen, and it value is changed such that it remains as a binary variable. The mutation procedure is described in Table 4.40.

Similar to that of the crossover operation, the resulting solution of mutation procedure may require “Repairing” due to the infeasibility which may have occurred due to the altering of a cell. However, at least one of tool availability and slot capacity constraints may become infeasible for the tool-machine pair as a result of changing a value to one, which has originally been zero before mutation on the chromosome.

Table 4.40. Pseudo code for mutation in proposed GA

…
(5.3) ***Apply mutation*** *on each child*
(5.3.1)If mutation condition is satisfied
(5.3.1.1)A tool is randomly selected from set of all tools
(5.3.1.2)A machine is randomly selected from set of all machines
(5.3.1.3)If the selected tool is already loaded on the selected machine in the child, that tool is unloaded from that machine, else that tool is loaded on that machine and infeasibilities due to mutation are repaired
…

In repairing procedure after mutation, slot capacities of machines are repaired for feasibility before tool availabilities. Repairing procedure of the slot capacity constraint violation is identical to that of the machine configuration exchange crossover procedure (see section 4.5.2) and the repairing of tool availability constraint is same with the repairing procedure of tool configuration exchange crossover (see section 4.5.1). Table 4.41 summarizes the requirement for a repairing procedure after the mutation applied on the cell in the intersection of row (tool type) i and column (machine) j.

Table 4.41. Repairing conditions for mutation

Original Value of *(i,j)*	Value of *(i,j)* after Mutation	Action
1	0	No Repairing Required
0	1	Repairing may be required

Table 4.42 illustrates the application of mutation procedure on a chromosome and represents the three alternative situations that can be encountered as a result of the mutation (Tool sizes are 1, tool availabilities and machines' slot capacities are equal to 2

and 3, respectively). The 3rd alternative shows that when t_1 is allocated on m_2, both tool availability and slot capacity constraints corresponding to cell (1, 2) become violated. Therefore, in order to maintain the feasibility of t_1's tool availability, it is unloaded from m_1 (randomly chosen among m_1 and m_3). Likewise, in order to maintain the feasibility of m_2's slot capacity, t_5 (randomly chosen among t_2, t_3 and t_5) is unloaded from m_2.

Table 4.42. Illustration of mutation

1st Alternative: 1→0 (No need to repair)

Child

	m_1	m_2	m_3
t_1	1	0	1
t_2	0	1	0
t_3	1	1	0
t_4	0	0	1
t_5	0	1	0
t_6	1	0	0

⇨

Child after Mutation

	m_1	m_2	m_3
t_1	1	0	1
t_2	0	1	0
t_3	1	1	0
t_4	0	0	1
t_5	0	0	0
t_6	1	0	0

2nd Alternative: 0→1 (No need to repair)

Child

	m_1	m_2	m_3
t_1	1	0	1
t_2	0	1	0
t_3	1	1	0
t_4	0	0	1
t_5	0	1	0
t_6	1	0	0

⇨

Child after Mutation

	m_1	m_2	m_3
t_1	1	0	1
t_2	0	1	0
t_3	1	1	0
t_4	0	0	1
t_5	0	1	1
t_6	1	0	0

3rd Alternative: 0→1 (Repairing required)

Child

	m_1	m_2	m_3
t_1	1	0	1
t_2	0	1	0
t_3	1	1	0
t_4	0	0	1
t_5	0	1	0
t_6	1	0	0

⇨

Child after Mutation

	m_1	m_2	m_3
t_1	**0**	1	1
t_2	0	1	0
t_3	1	1	0
t_4	0	0	1
t_5	0	**0**	0
t_6	1	0	0

4.7. Improvement

Another genetic operator, "Improvement Procedure", is developed specifically for the problem, and is applied on children that have gone through crossover and mutation with corresponding probabilities. The improvement has two parameters: improvement period and improvement probability. Improvement procedure is as follows:

Table 4.43. Pseudo code for improvement in proposed GA

```
…
(5.4) Improve children if improvement conditions are satisfied
    (5.4.1) For each child, if the generation is in the improvement
            period and improvement condition is satisfied
            improvement procedure is applied
…
```

The chromosomes in the initial population are generated by complete utilization of tool magazines, meaning that every tool in the FMS is allocated to a machine and no tool magazine has any tool slot left empty in the initial population. However, the genetic operations applied over generations tend to decrease the number of tool allocations (number of 1's) in chromosome. The reason for this phenomenon is changing 1's to 0's in order to repair the infeasibilities that come up as a result of genetic operations. Table 4.38, Table 4.39 and Table 4.42 show such situations. In order to compensate for the effects of this worsening in fitness values, two types of improvement procedures are developed and tested to see their effects on the performance of the GA.

- Improvement by Allocating a Single Idle Tool
- Improvement by Allocating All Idle Tools

4.7.1. Improvement by Allocating a Single Idle Tool

Due to genetic operations applied over generations, number of 1's in a chromosome which denotes the number of all tools in the FMS decreases. This phenomenon is interpreted as some tools becoming idle because of not being allocated to any of the machine's tool magazines.

Allocating a tool on a machine is expected to increase the machine and route flexibilities of the FMS. Improvement by allocating a single idle tool is done as shown in Table 4.44:

Table 4.44. Pseudo code for improvement by allocating a single idle tool

List of Idle Tools *denotes the tool types whose tool availability constraints are not binding in the chromosome under consideration*
List of Available Machines *for a tool denotes the machines whose tool magazines have enough slot capacities for that tool to be allocated on*

(1) Determine the List of Idle Tools for the chromosome
(2) If there are at least one tool in the List of Idle Tools, choose a tool randomly from List of Idle Tools, else STOP
(3) Determine the List of Available Machines to load the chosen tool
(4) If there are at least one machine in the List of Available Machines, choose a machine randomly from the List of Available Machines, else discard the chosen tool from the List of Idle Tools and go to (2)
(5) Allocate the chosen tool to the chosen machine

4.7.2. Improvement by Allocating All Idle Tools

The initial population is generated such that all tools are completely allocated on all machines. As an alternative to making a single allocation as described in section 4.7.1,

making an idle tool allocation as long as possible is considered as a second alternative. The allocation procedure is presented in Table 4.45.

Table 4.45. Pseudo code for improvement by allocating all idle tools

List of Idle Tools *denotes the tool types whose tool availability constraints are not binding in the chromosome under consideration*
List of Available Machines *for a tool denotes the machines whose tool magazines have enough slot capacities for that tool to be allocated on*

(1) Determine the List of Idle Tools for the chromosome
(2) If there are at least one tool in the List of Idle Tools, choose a tool randomly from List of Idle Tools, else STOP
(3) Determine the List of Available Machines to load the chosen tool
(4) If there are at least one machine in the List of Available Machines, choose a machine randomly from the List of Available Machines, else discard the chosen tool from the List of Idle Tools and go to (2)
(5) Allocate the chosen tool to the chosen machine and go to (2)

The only difference of the above procedure from the one in the previous section is step (5) which enables the procedure to repeat itself as long as possible.

4.7.3. Parameters for Improvement Procedure

Two parameters for the improvement procedure developed for the tool allocation problem are the improvement period and improvement probability.

Improvement period represents the generations where improvement procedure is allowed to be applied on children. For instance if the improvement period is 5, the children that will form up the generations 5, 10, 15, … are allowed to be improved.

A child on a generation to be improved is not necessarily required to go through an improvement stage. In order to maintain the diversity among the chromosomes, it may be

helpful to leave the chromosomes as they are to fully search the solution space. For this purpose, the improvement procedure is developed as a probabilistic process. The children in an improvement generation are improved with an improvement probability.

The two types of improvement procedures and parameters settings (improvement period and improvement probability) will be done by analyzing the experimental results.

4.8. Advancing to the Next Generation

The determination of the next population follows an elitist strategy. The individual with the best fitness value is preserved for the next population. The rest of the population is filled by selecting individuals from the current population, and applying genetic operations on these individuals.

The mates are selected as couples from the population. These mates go through crossover, mutation and improvement stages with relevant probabilities. If the labels of two children that have gone through genetic operations become identical, only one of them is allowed to survive and inserted to the next population due to the no duplicates strategy.

The number of children advancing to the next generation is allowed to exceed the predetermined population size. The reason of this situation and its remedy is explained in section 4.3.

Table 4.46. Pseudo code for advancing to the next generation in proposed GA

…

(6) ***Advance the population*** *to the next generation's population, and go to (2)*

5. NUMERICAL STUDY

In order to choose the best genetic operators among the alternatives and to determine the best settings for the GA parameters, a series of experiments need to be done. This section introduces generation of the test problems used in experimentations, experimentation related to determining GA strategies, experimentation related to fine tuning of GA parameters and comparison of the proposed GA approach with an exact algorithm.

5.1. Generation of Test Problems

Ten alternative test problem layouts are generated to be used as test-beds throughout the experimentation process. These problem layouts have also been used while determining the heuristic algorithm for fitness evaluation in section 4.3.6. Details of these ten problems are presented in APPENDIX A. Table 5.1 gives an idea about the characteristic of the problems. The problems are specified such that they would represent various characteristics.

Table 5.1. Characteristics of the test problems

Problem	Number of Part Types	Total Number of Processing Alternatives	Number of Operations	Number of Tools	Number of Machines
P1	3	7	20	12	3
P2	3	8	20	11	4
P3	4	9	24	14	5
P4	4	12	31	14	7
P5	5	13	36	14	10
P6	5	16	45	16	9
P7	7	20	60	13	9
P8	10	26	77	18	6
P9	10	32	92	22	6
P10	8	26	76	21	7

Test problems are defined such that the number of part types and the number of parts from each part type differ from each other. By doing this, workloads caused by the part types do not show a standard pattern. Workloads due to some of the part types are close or identical to each other, while others impose imbalance. This enables to see the behavior of procedure in different cases.

For problems where number of part types is equal, processing alternatives are altered in a way such that extra routes are included for a part type or existing operation sequences are modified. Operation processing times are assigned such that various effects could be observed. For some part types, the total route processing times are very close to each other, while for others, they are significantly different. This shows the operation of the developed heuristic and GA algorithm in various conditions.

The list of operations that each tool type can process is specified such that most of the tools are capable of processing several operations. However, in all problems there exist at least one tool that can process only a single operation to simulate the presence of a bottleneck in terms of tool requirement. Tool availabilities are chosen such that the average expected workloads on tools can be met by the existing number of tools in the system.

Number of identical machines in the FMS and their standard available processing times differ from problem to problem. Each tool type's tool size and machine's slot capacities are set simultaneously such that tool sizes differ from each other and the slot magazines' total capacity is equal to the total slot coverage of all the tools available in the problem.

5.2. Selection of GA strategies

Alternatives for calculation of selection probabilities, crossover and improvement procedures have been introduced in sections 4.4, 4.5 and 0. These alternatives are summarized in Table 5.2.

In order to choose the best combination of selection, crossover and improvement alternatives, a series of experiments have to be done. The first six test problems presented

in APPENDIX A are selected for the experimentation process. The combination that shows the best average performance over these problems will be chosen and the strategies for the developed GA will be determined.

Table 5.2. Alternative GA strategies

Factor Levels	Selection Probability Calculation Alternatives
S1	Fitness-Based Selection Probability Calculation
S2	Ranking-Based Selection Probability Calculation
Factor Levels	Crossover Alternatives
C1	Tool Configuration Exchange Crossover
C2	Machine Configuration Exchange Crossover
Factor Levels	Improvement Alternatives
I1	Improvement by Allocating a Single Idle Tool
I2	Improvement by Allocating All Idle Tools

However, in order to carry out experiments at this stage, the GA parameter levels to be used in runs has to be determined. These parameters are set as shown in Table 5.3. One should note that these parameter settings are preliminary values and will be further analyzed.

Test problems 1, 2 and 3 are considered to be relatively smaller than the others. For this reason, population sizes are set as 50 for these problems. The others (test problems 4, 5 and 6) use 100 as the population size.

Table 5.3. Preliminary GA parameters used in determining GA strategies

GA Parameter	Preliminary Value
Population Size	50 (P1, P2, P3); 100 (P4, P5, P6)
Crossover Probability	0.8
Mutation Probability	0.1
Improvement Period	5
Improvement Probability	0.5
Maximum Number of Generations	30

For all 8 (=2^3) combination alternatives five replications are executed. Each run returns a minimum fitness value that corresponds to that run's best solution. The minimum fitness values for a combination's replications are averaged. The results are presented in Table 5.4.

Table 5.4. Average of five replications' best fitness values

Combination	P1	P2	P3	P4	P5	P6
S1-C1-I1	1.650	7.041	4.930	8.764	6.787	7.463
S1-C1-I2	1.386	7.041	5.192	8.555	6.454	6.847
S1-C2-I1	1.748	7.044	5.286	8.969	6.126	7.457
S1-C2-I2	1.649	7.048	4.247	8.310	6.091	7.050
S2-C1-I1	1.842	7.037	4.463	8.362	6.456	6.837
S2-C1-I2	1.207	7.041	5.051	8.306	6.260	7.037
S2-C2-I1	1.925	7.044	5.407	8.775	5.948	7.454
S2-C2-I2	1.566	7.037	4.579	8.126	5.647	7.042
Best of Averages	1.207	7.037	4.247	8.126	5.647	6.837

The performance of a combination on a test problem is evaluated by calculating the per cent gap of that combination's output value (given in Table 5.4) from the best (lowest) value attained among all combinations for that problem. In other words, for each problem the minimum average fitness value attained is determined and each combination's per cent gap from that value is calculated by the ratio, differences in average fitness values divided by the best fitness value attained. Table 5.5 depicts the results of this analysis.

Table 5.5. Per cent gaps from best average fitness values for GA strategies

Combination	P1	P2	P3	P4	P5	P6	Average % Gap
S1-C1-I1	36.77%	0.05%	16.10%	7.85%	20.17%	9.15%	15.01%
S1-C1-I2	14.83%	0.05%	22.27%	5.27%	14.28%	0.15%	9.48%
S1-C2-I1	44.84%	0.09%	24.48%	10.36%	8.47%	9.08%	16.22%
S1-C2-I2	36.68%	0.14%	0.00%	2.26%	7.86%	3.12%	8.34%
S2-C1-I1	52.65%	0.00%	5.09%	2.90%	14.32%	0.00%	12.49%
S2-C1-I2	0.00%	0.05%	18.93%	2.21%	10.85%	2.93%	5.83%
S2-C2-I1	59.50%	0.09%	27.32%	7.98%	5.33%	9.03%	18.21%
S2-C2-I2	29.75%	0.00%	7.83%	0.00%	0.00%	3.00%	6.76%

The rightmost column of Table 5.5 shows that S2-C1-I2 combination is the best performing one among others with average per cent gap value of 5.83 per cent. The second best combination is S2-C2-I2 with 6.76 per cent.

By looking at these results, it is thought that applying a hybrid crossover strategy (a combination of C1 and C2) would be a good idea. For this reason, the two crossover strategies (C1 and C2) are combined within GA.

When the crossover condition is satisfied, a pseudo random number determines which crossover strategy to apply on the mate. This situation introduces a new parameter to the GA, namely P_{C1}, which is equal to the probability of applying C1 (tool exchange crossover) on a mate. In order to choose the best level of P_{C1}, a series of experiments are done (using the GA parameters presented in Table 5.4). Note that P_{C2} will equal to (1 – P_{C1}) in implementations. Hybrid crossover approach is summarized in Table 5.6.

Table 5.6. Pseudo code for hybrid crossover operator in proposed GA

…
(5.2) ***Apply crossover*** *on two mates to form two children*
(5.2.1) If crossover condition is satisfied
(5.2.1.1) If a pseudo random number generated is less than 0.8, apply tool exchange crossover, else apply machine exchange crossover to form the two selected mates into two children
(5.2.1.2) If crossover has caused any violation in feasibility conditions, a repairing procedure is applied for each child
…

The results, presented in Table 5.7, represents the average of five replications for each (problem, P_{C1}) pair.

If the bottom rows (best averages) of Table 5.4 and Table 5.7 are compared, it can easily be seen that introduction of C3 has accomplished to reach better results. Moreover the two tables (Table 5.4 and Table 5.7) are combined and corresponding per cent gap values from the overall best averages are calculated. The results are displayed in Table 5.8 reveals that GA strategy denoted by S2-C3-I2 with P_{C1}=0.8 turn out to be the best among all with per cent gap value of 6.19 per cent. Table 5.9 summarizes the GA strategies chosen as a result of the analysis.

Table 5.7. Average of five replications' best fitness values

S2-C3-I2	P1	P2	P3	P4	P5	P6
P_{C1}=0.1	1.471	7.044	4.516	7.939	5.640	6.637
P_{C1}=0.2	1.289	7.041	5.431	7.934	6.039	6.441
P_{C1}=0.3	1.567	7.041	5.734	8.109	5.641	7.044
P_{C1}=0.4	1.564	7.041	4.927	8.140	5.652	6.852
P_{C1}=0.5	1.207	7.037	5.459	7.753	5.919	6.843
P_{C1}=0.6	1.561	7.041	4.929	7.874	5.246	7.037
P_{C1}=0.7	1.650	7.037	4.868	8.294	6.052	7.246
P_{C1}=0.8	1.468	7.041	4.038	8.114	5.645	6.646
P_{C1}=0.9	1.387	7.037	4.596	8.115	5.850	6.839
Best Averages	1.207	7.037	4.038	7.753	5.246	6.441

Table 5.8. Per cent gaps from best average fitness values for GA strategies including hybrid crossover strategy

Combination	P1	P2	P3	P4	P5	P6	Average % Gap
S1-C1-I1	36.77%	0.05%	22.10%	13.05%	29.37%	15.86%	19.53%
S1-C1-I2	14.83%	0.05%	28.60%	10.35%	23.03%	6.30%	13.86%
S1-C2-I1	44.84%	0.09%	30.92%	15.69%	16.78%	15.78%	20.68%
S1-C2-I2	36.68%	0.14%	5.17%	7.19%	16.12%	9.45%	12.46%
S2-C1-I1	52.65%	0.00%	10.52%	7.86%	23.07%	6.14%	16.71%
S2-C1-I2	0.00%	0.05%	25.09%	7.14%	19.33%	9.25%	10.14%
S2-C2-I1	59.50%	0.09%	33.90%	13.19%	13.39%	15.73%	22.63%
S2-C2-I2	29.75%	0.00%	13.41%	4.82%	7.66%	9.32%	10.83%

Table 5.8 continued

Combination	P1	P2	P3	P4	P5	P6	Average % Gap
S2-C3-I2, P_{C1}=0.1	21.94%	0.09%	11.85%	2.41%	7.52%	3.04%	7.81%
S2-C3-I2, P_{C1}=0.2	6.85%	0.05%	34.50%	2.34%	15.12%	0.00%	9.81%
S2-C3-I2, P_{C1}=0.3	29.84%	0.05%	42.01%	4.60%	7.52%	9.36%	15.56%
S2-C3-I2, P_{C1}=0.4	29.58%	0.05%	22.01%	5.00%	7.75%	6.38%	11.80%
S2-C3-I2, P_{C1}=0.5	0.00%	0.00%	35.20%	0.00%	12.83%	6.25%	9.05%
S2-C3-I2, P_{C1}=0.6	29.41%	0.05%	22.07%	1.57%	0.00%	9.26%	10.39%
S2-C3-I2, P_{C1}=0.7	36.77%	0.00%	20.56%	6.99%	15.36%	12.50%	15.36%
S2-C3-I2, P_{C1}=0.8	21.68%	0.05%	0.00%	4.66%	7.60%	3.18%	6.19%
S2-C3-I2, P_{C1}=0.9	14.92%	0.00%	13.82%	4.68%	11.51%	6.18%	8.52%

Table 5.9. GA strategies to be applied for the tool allocation problem

GA Strategy	Chosen Strategy
Selection Probability Calculation	Ranking-Based Selection Probability Calculation
Crossover	Hybrid Crossover Operator
Improvement	Improvement by Allocating All Idle Tools

5.3. Fine tuning of GA Parameters

In this section, experiments are carried out in order to fine tune the preliminary GA parameter values set previously in section 5.2. The fine tuning procedure for GA parameters is decided to be done in the following order:

(1) Population Size (N)
(2) Crossover Probability (PX)
(3) Mutation Probability (PM)
(4) Improvement Probability (PI)
(5) Improvement Period (IP)
(6) Maximum Number of Generations (G)

The initial values of the above parameters have been introduced in Table 5.3.

5.3.1. Fine Tuning of Population Size

Two more alternatives are compared against the population size levels introduced in Table 5.10. The first alternative is determining the population size as the multiplication of number of tool types and number of machines for each problem. Such a population size determination method could be useful in a number of ways.

First of all, the population size could have a generalized formula "t×m", where the number of tool types is denoted by t, and the number of machines is denoted by m (e.g. N = 12×3 = 36 for test problem layout #1). In fact, "t×m" is the number of cells in the matrix-based solution representation for a problem. If this approach turns out to be well-performing, it can be generalized and used in determining the population size in all kinds of problems.

The second alternative is determined by expanding the preliminary values by 25 per cent. The resulting population size levels for each problem and the corresponding average fitness values over five replications are presented in Table 5.11.

Experimentation results have been used to calculate the per cent gap values for each population size level. Results are presented in Table5.12.

It is observed that N2 (50 for P1, P2, P3; 100 for P4, P5, P6) turns out to have the best average per cent gap from the best average fitness value attained overall. N2 is set for the rest of the experiments. For test problems P7 to P10, 100 will be used as population size.

Table 5.10. Alternatives for population size (N)

Factor Level	Population Size
N1	t×m
N2	50 (P1, P2, P3); 100 (P4, P5, P6)
N3	60 (P1, P2, P3); 120 (P4, P5, P6)

Table 5.11. Experimentation results for fine tuning of population size

Alternatives	P1	P2	P3	P4	P5	P6
N1	36	44	70	98	140	144
Average Fitness Value	1.647	7.041	4.255	8.141	5.863	6.637
N2	50	50	50	100	100	100
Average Fitness Value	1.468	7.041	4.038	8.114	5.645	6.646
N3	60	60	60	120	120	120
Average Fitness Value	1.297	7.037	4.454	7.952	6.045	7.238
Best of Averages	1.297	7.037	4.038	7.952	5.645	6.637

Table 5.12. Per cent gaps from best average fitness values for population size alternatives

Population Size Alternatives	P1	P2	P3	P4	P5	P6	Average % Gap
N1	36	44	70	98	140	144	Average
% Gaps	27.02%	0.05%	5.38%	2.38%	3.87%	0.00%	6.45%
N2	50	50	50	100	100	100	Average
% Gaps	13.22%	0.05%	0.00%	2.04%	0.00%	0.14%	2.57%
N3	60	60	60	120	120	120	Average
% Gaps	0.00%	0.00%	10.31%	0.00%	7.10%	9.06%	4.41%

5.3.2. Fine Tuning of Crossover Probability

In addition to the preliminary value of 0.8, various levels for the crossover probability has been implemented. Table 5.13 presents the six levels that have been tested against each other. The run results have been analyzed similar to the analysis of GA strategies selection and fine tuning of population size. Factor levels' average fitness values and per cent gaps from the best attainable average for each test problem are presented in Table 5.14 and Table 5.15, respectively.

Table 5.13. Alternatives for crossover probability (PX)

Factor Level	Crossover Probability
PX1	0.5
PX2	0.6
PX3	0.7
PX4	0.8
PX5	0.9
PX6	1.0

Table 5.14. Experimentation results for fine tuning of crossover probability

N	50	50	50	100	100	100
Crossover Probability Levels	P1	P2	P3	P4	P5	P6
PX1	1.387	7.041	5.332	7.919	5.684	6.446
PX2	1.290	7.044	4.576	8.101	5.449	6.840
PX3	1.468	7.037	4.524	8.316	6.058	6.859
PX4	1.468	7.041	4.038	8.114	5.645	6.646
PX5	1.649	7.041	4.524	8.122	5.242	7.048
PX6	1.470	7.044	4.057	8.161	5.851	7.049
Best of Averages	1.290	7.037	4.038	7.919	5.242	6.446

Table 5.15. Per cent gaps from best average fitness values for crossover probability levels

N	50	50	50	100	100	100	
Crossover Probability Levels	P1	P2	P3	P4	P5	P6	Average % Gap
PX1	7.47%	0.05%	32.06%	0.00%	8.43%	0.00%	8.00%
PX2	0.00%	0.09%	13.32%	2.30%	3.95%	6.11%	4.30%
PX3	13.79%	0.00%	12.05%	5.01%	15.57%	6.41%	8.80%
PX4	13.79%	0.05%	0.00%	2.46%	7.69%	3.10%	4.51%
PX5	27.82%	0.05%	12.04%	2.56%	0.00%	9.34%	8.64%
PX6	13.95%	0.09%	0.47%	3.06%	11.63%	9.36%	6.43%

The results show that PX2 (i.e. 0.6) has turned out to be the best performing level for the crossover probability. This level will be fixed for the rest of the experimentation process.

5.3.3. Fine Tuning of Mutation Probability

The alternative probability levels to be tested for the mutation operator are presented in Table 5.16. The preliminary value of 0.1 turns out to be better performing than the other levels. The run results have been analyzed similar previous fine tuning analysis. Average fitness values and per cent gaps for each test problem are presented in Table 5.17 and Table 5.18, respectively.

Table 5.16. Alternatives for mutation probability (PM)

Factor Level	Mutation Probability
PM1	0.05
PM2	0.1
PM3	0.2
PM4	0.5
PM5	1.0

Table 5.17. Experimentation results for fine tuning of mutation probability

PX	0.6	0.6	0.6	0.6	0.6	0.6
N	50	50	50	100	100	100
Mutation Probability Levels	P1	P2	P3	P4	P5	P6
PM1	1.469	7.044	4.512	7.696	6.068	6.838
PM2	1.290	7.044	4.576	8.101	5.449	6.840
PM3	1.828	7.037	4.035	7.927	6.009	7.037
PM4	1.287	7.044	4.252	7.935	6.316	7.054
PM5	1.742	7.041	4.461	8.363	5.954	7.243
Best of Averages	1.287	7.037	4.035	7.696	5.449	6.838

Table 5.18. Per cent gaps from best average fitness values for mutation probability levels

PX	0.6	0.6	0.6	0.6	0.6	0.6	
N	50	50	50	100	100	100	
Mutation Probability Levels	P1	P2	P3	P4	P5	P6	Average % Gap
PM1	14.14%	0.09%	11.82%	0.00%	11.36%	0.00%	6.24%
PM2	0.24%	0.09%	13.39%	5.26%	0.00%	0.03%	3.17%
PM3	42.03%	0.00%	0.00%	3.00%	10.28%	2.92%	9.70%
PM4	0.00%	0.09%	5.38%	3.11%	15.91%	3.16%	4.61%
PM5	35.37%	0.05%	10.56%	8.67%	9.27%	5.92%	11.64%

5.3.4. Fine Tuning of Improvement Probability

Improvement procedure has been applied on chromosomes with a probability of 0.5 so far. In this section, alternatives for this probability that may result with better performance are searched.

The alternative probability levels to be tested are presented in Table 5.19. Once again, the preliminary value of a GA parameter turns out to be better performing than the other levels.

Average fitness values and per cent gaps for each test problem are presented in Table 5.20 and Table 5.21, respectively.

Table 5.19. Alternatives for improvement probability (PI)

Factor Level	Improvement Probability
PI1	0.3
PI2	0.4
PI3	0.5
PI4	0.6
PI5	0.7

Table 5.20. Experimentation results for fine tuning of improvement probability

PM	0.1	0.1	0.1	0.1	0.1	0.1
PX	0.6	0.6	0.6	0.6	0.6	0.6
N	50	50	50	100	100	100
Improvement Probability Levels	P1	P2	P3	P4	P5	P6
PI1	1.290	7.044	4.516	8.567	5.847	6.640
PI2	1.927	7.037	5.140	8.105	5.041	6.843
PI3	1.290	7.044	4.576	8.101	5.449	6.840
PI4	1.650	7.044	4.645	8.114	5.512	6.844
PI5	1.469	7.037	4.718	7.755	5.647	6.847
Best of Averages	1.290	7.037	4.516	7.755	5.041	6.640

Table 5.21. Per cent gaps from best average fitness values for improvement probability levels

PM	0.1	0.1	0.1	0.1	0.1	0.1	
PX	0.6	0.6	0.6	0.6	0.6	0.6	
N	50	50	50	100	100	100	
Improvement Probability Levels	P1	P2	P3	P4	P5	P6	Average % Gap
PI1	0.00%	0.09%	0.00%	10.47%	15.98%	0.00%	4.42%
PI2	49.32%	0.00%	13.82%	4.51%	0.00%	3.05%	11.78%
PI3	0.00%	0.09%	1.33%	4.46%	8.09%	3.01%	2.83%
PI4	27.90%	0.09%	2.86%	4.62%	9.34%	3.06%	7.98%
PI5	13.87%	0.00%	4.49%	0.00%	12.01%	3.10%	5.58%

5.3.5. Fine Tuning of Improvement Period

The frequency of the improvement procedure has been preliminarily chosen to be once in five generations. Two alternative levels for this period are presented in Table 5.22. The preliminary level of improvement period also performs the best among others. Average fitness values and per cent gaps for each test problem for the levels are presented in Table 5.23 and Table 5.24.

Table 5.22. Alternatives for improvement period (IP)

Factor Level	Improvement Period
IP1	3
IP2	5
IP3	7

Table 5.23. Experimentation results for fine tuning of improvement period

PI	0.5	0.5	0.5	0.5	0.5	0.5
PM	0.1	0.1	0.1	0.1	0.1	0.1
PX	0.6	0.6	0.6	0.6	0.6	0.6
N	50	50	50	100	100	100
Improvement Period Levels	P1	P2	P3	P4	P5	P6
IP1	1.562	7.037	4.704	8.096	5.848	6.442
IP2	1.290	7.044	4.576	8.101	5.449	6.840
IP3	1.745	7.044	4.837	8.125	5.647	6.642
Best of Averages	1.290	7.037	4.576	8.096	5.449	6.442

Table 5.24. Per cent gaps from best average fitness values for improvement period levels

PI	0.5	0.5	0.5	0.5	0.5	0.5	
PM	0.1	0.1	0.1	0.1	0.1	0.1	
PX	0.6	0.6	0.6	0.6	0.6	0.6	
N	50	50	50	100	100	100	
Mutation Probability Levels	P1	P2	P3	P4	P5	P6	Average % Gap
IP1	21.10%	0.00%	2.80%	0.00%	7.32%	0.00%	5.20%
IP2	0.00%	0.09%	0.00%	0.07%	0.00%	6.17%	1.06%
IP3	35.21%	0.09%	5.71%	0.36%	3.63%	3.09%	8.02%

5.3.6. Fine Tuning of Maximum Number of Generations

The maximum number of generations for GA runs has been set as 30 generations in preliminary runs. In this stage, the performance of the algorithm will be checked in increased values of maximum number of generations. Increasing the number of generations

that GA is executed would directly affect the total runtime of the algorithm. However, if the results turn out to be promising, increased levels may be considered. The alternative levels for this period are presented in Table 5.25. Average fitness values and per cent gaps for each test problem for the levels are presented in Table 5.26 and Table 5.27.

Table 5.25. Alternatives for maximum number of generations (G)

Factor Level	Improvement Probability
G1	30
G2	40
G3	50

Table 5.26. Experimentation results for fine tuning of maximum number of generations

IP	5	5	5	5	5	5
PI	0.5	0.5	0.5	0.5	0.5	0.5
PM	0.1	0.1	0.1	0.1	0.1	0.1
PX	0.6	0.6	0.6	0.6	0.6	0.6
N	50	50	50	100	100	100
Maximum Number of Generations	P1	P2	P3	P4	P5	P6
G1	1.290	7.044	4.576	8.101	5.449	6.840
G2	1.289	7.041	4.576	7.887	5.446	6.833
G3	1.289	7.037	4.372	7.882	5.443	6.633
Best of Averages	1.289	7.037	4.372	7.882	5.443	6.633

Table 5.27. Per cent gaps from best average fitness values for maximum number of generations levels

IP	5	5	5	5	5	5	
PI	0.5	0.5	0.5	0.5	0.5	0.5	
PM	0.1	0.1	0.1	0.1	0.1	0.1	
PX	0.6	0.6	0.6	0.6	0.6	0.6	
N	50	50	50	100	100	100	
Mutation Probability Levels	P1	P2	P3	P4	P5	P6	Average % Gap
G1	0.08%	0.09%	4.66%	2.78%	0.10%	3.12%	1.80%
G2	0.00%	0.05%	4.66%	0.06%	0.05%	3.02%	1.31%
G3	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

The maximum number of generation for all test problems is set to 50. Values beyond this level are considered to be prohibitive in terms of computation times. The finalized set of GA parameters is presented in Table 5.28.

Table 5.28. The finalized set of GA parameters

GA Parameter	Finalized Value
Population Size	50 (P1:P3); 100 (P4:P10)
Crossover Probability	0.6
Mutation Probability	0.1
Improvement Period	5
Improvement Probability	0.5
Maximum Number of Generations	50

5.4. Final Results

The finalized set GA is applied on the rest of the test problems (P7 to P10). The whole set of test problems are also run by the CPLEX algorithm. CPLEX runs are limited with one day (86,400 seconds) of computation time. If the algorithm does not find the optimal solution until then, the feasible solution with the best objective value attained so far together with its per cent gap from the best possible LP relaxation objective is recorded. GA runs are carried out with five replications. The results are given in Table 5.29, where the rightmost column denotes the average per cent gaps from the best attained CPLEX objective.

For P9 and P10, CPLEX algorithm was not able to find a feasible solution within the computation time limit. For this reason, per cent gap values are not computed for these problems.

All problems and procedures are run using a P4 1.6 processor. The corresponding computation times for GA and CPLEX runs are presented in Table 5.30.

Table 5.29. Final results of the GA and CPLEX algorithm

Problem Layout	Total Number of Parts	CPLEX Objective Value (% gap)	GA Best Objective	GA Average Objective	GA Worst Objective	Average % Gap
P1	28	1.100	1.110	1.289	2.005	17.20%
P2	38	7.006	7.037	7.037	7.037	0.45%
P3	35	3.678 (0.06%)	4.033	4.372	5.036	18.87%
P4	48	6.014 (16.86%)	7.097	7.882	8.095	31.07%
P5	46	4.010 (25.20%)	5.038	5.443	6.050	35.74%
P6	50	6.006 (16.74%)	6.043	6.633	7.030	10.44%
P7	48	2.119 (47.57%)	2.265	3.147	3.380	48.50%
P8	76	17.211 (4.04%)	19.127	19.360	19.582	12.49%
P9	76	-	2.019	2.476	3.034	-
P10	71	-	2.324	3.235	3.509	-

Table 5.30. Computation times of the GA and CPLEX algorithm

Problem	GA Average Computation Time (sec)	CPLEX Computation Time (sec)
P1	96	4
P2	123	208
P3	192	86,400
P4	771	86,400
P5	1774	86,400
P6	2130	86,400
P7	2660	86,400
P8	3884	86,400
P9	7163	86,400
P10	5120	86,400

The analysis on the performance reveals that the GA approach developed as a result of this study is promising. The average per cent gaps for all problems are accepted to be satisfactory. As already mentioned, the objective value is a representation of the number of uncompleted parts of the problem. The worst average per cent gap is attained in problem #7 results. Even in that case, it can be easily seen that as opposed to the two remaining parts for the CPLEX algorithm, the GA procedure leaves three unfinished parts out of initial total of 48 parts. Due to the complexity of the problem, there have been times that the CPLEX algorithm was not able to attain any feasible solutions.

6. CONCLUSIONS

This thesis introduces a GA to the tool allocation problem, which is a setup problem in an FMS and is influential on the overall system performance. The tool allocation problem as defined in this study involves allocation of tools into the limited tool magazines of a set of machines in an FMS in order to produce a set of part batches. The complexity of the tool allocation problem prevents obtaining optimal solutions in reasonable amount of time. Hence, it would be a necessity to apply efficient heuristics which provides sufficiently good solutions in affordable amount of time. The proposed GA approach is based on acquiring the tool allocation from a matrix-based chromosome and loading the operations of part types by running a heuristic algorithm. There are lots of heuristic approaches presented in the literature dealing with tool allocation problem. However, at least to our knowledge, none of these approaches try to utilize flexibilities related to routing. The flexibility which stems from the part types' nature is most of the time not considered in proposed algorithms so far. This study, in this respect, is an important one which tries to fill this gap in the field. In addition to the absence of introducing such flexibilities, most of the work regarding to tool allocation ignores order splitting and solves tooling problem by first solving operation loading problem. The comparison of the GA with an exact algorithm on test problems reveals that the proposed GA approach attains satisfactory results in terms of both average objective values and computation time.

APPENDIX A: TEST PROBLEM LAYOUTS

Table A.1. Test problem #1 layout

Test Problem #1 Layout		
Part Type	Demand	Processing Alternatives (Unit Processing Times)
A	10	opr1A (4) - opr2A (4) - opr3A (8)
		opr4A (5) - opr5A (4) - opr6A (5)
B	11	opr1B (7) - opr2B (8)
		opr3B (7) - opr4B (6) - opr5B (4)
		opr6B (5) - opr7B (8)
C	7	opr1C (6) - opr2C (7) - opr3C (4)
		opr4C (5) - opr5C (7) - opr6C (5) - opr7C (6)
Tool Type (Size)	Availability	Tool's Operations
t1 (1)	1	opr1A
t2 (1)	1	opr2A, opr3B, opr7B
t3 (1)	1	opr3A, opr6B
t4 (1)	1	opr4A
t5 (1)	1	opr5A, opr5B, opr7C
t6 (1)	1	opr6A, opr2B
t7 (1)	1	opr1C
t8 (1)	1	opr1B, opr4B
t9 (1)	1	opr2C, opr6C
t10 (1)	1	opr5C
t11 (1)	1	opr3C
t12 (1)	1	opr4C
Machines		
Number of Machines		3
Available Processing Times		130
Slot Capacities		4

Table A.2. Test problem #2 layout

Test Problem #2 Layout		
Part Type	Demand	Processing Alternatives (Unit Processing Times)
A	17	opr1A (5) - opr2A (7) - opr3A (8)
		opr4A (4) - opr5A (4)
B	12	opr1B (8) - opr2B (8) - opr3B (8)
		opr4B (4) - opr5B (4)
		opr6B (8) - opr7B (8)
C	9	opr1C (6) - opr2C (7) - opr3C (4)
		opr4C (5) - opr5C (7)
		opr6C (5) - opr7C (6) - opr8C (4)
Tool Type (Size)	Availability	Tool's Operations
t1 (1)	1	opr1A, opr7C
t2 (1)	2	opr2A, opr3B, opr7B
t3 (2)	2	opr3A, opr6B
t4 (2)	1	opr4A, opr4C
t5 (1)	1	opr5A, opr5B
t6 (1)	1	opr2B
t7 (1)	1	opr1C
t8 (1)	1	opr1B, opr4B
t9 (1)	1	opr2C, opr6C, opr8C
t10 (1)	1	opr5C
t11 (1)	1	opr3C
Machines		
Number of Machines		4
Available Processing Times		80
Slot Capacities		4

Table A.3. Test problem #3 layout

Test Problem #3 Layout		
Part Type	Demand	Processing Alternatives (Unit Processing Times)
A	5	opr1A (6) - opr2A (7)
		opr3A (6) - opr4A (4) - opr5A (6)
B	12	opr1B (6) - opr2B (8)
		opr3B (7) - opr4B (6)
C	9	opr1C (6) - opr2C (4) - opr3C (6)
		opr4C (7) - opr5C (8) - opr6C (5)
D	9	opr1D (8) - opr2D (5) - opr3D (5)
		opr4D (5) - opr5D (6) - opr6D (4) - opr7D (5)
		opr8D (5) - opr9D (8)
Tool Type (Size)	Availability	Tool's Operations
t1 (2)	1	opr1A, opr8D
t2 (1)	1	opr2A, opr7D
t3 (1)	1	opr3A, opr6D
t4 (1)	1	opr4A, opr4D
t5 (2)	1	opr5A, opr3D, opr5D
t6 (1)	1	opr1B, opr2D
t7 (2)	1	opr2B, opr1D
t8 (1)	1	opr3B
t9 (2)	1	opr4B, opr6C
t10 (1)	1	opr1C
t11 (2)	1	opr2C
t12 (1)	1	opr3C
t13 (1)	1	opr4C
t14 (2)	1	opr5C, opr9D
Machines		
Number of Machines		5
Available Processing Times		90
Slot Capacities		4

Table A.4. Test problem #4 layout

Test Problem #4 Layout		
Part Type	Demand	Processing Alternatives (Unit Processing Times)
A	9	opr1A (4) - opr2A (8) - opr3A (5)
		opr4A (4) - opr5A (8)
		opr6A (6) - opr7A (8) - opr8A (4)
B	15	opr1B (6) - opr2B (6) - opr3B (4)
		opr4B (6) - opr5B (7)
		opr6B (8) - opr7B (5)
C	14	opr1C (6) - opr2C (5) - opr3C (5)
		opr4C (6) - opr5C (6)
D	10	opr1D (5) - opr2D (6)
		opr3D (4) - opr4D (5) - opr5D (8) - opr6D (6)
		opr7D (8) - opr8D (5) - opr9D (7)
		opr10D (6) - opr11D (7)
Tool Type (Size)	Availability	Tool's Operations
t1 (2)	1	opr1A, opr7B, opr2D
t2 (2)	2	opr2A, opr1C
t3 (2)	1	opr3A, opr2C
t4 (1)	1	opr4A, opr3C
t5 (2)	2	opr5A, opr4C, opr8D
t6 (2)	2	opr6A, opr5C, opr9D
t7 (1)	1	opr7A, opr10D
t8 (2)	1	opr8A, opr11D
t9 (1)	1	opr1B, opr7D
t10 (2)	1	opr2B
t11 (2)	1	opr3B, opr6D
t12 (1)	1	opr4B, opr1D, opr5D
t13 (1)	1	opr5B, opr4D
t14 (1)	1	opr6B, opr3D
Machines		
Number of Machines		7
Available Processing Times		75
Slot Capacities		4

Table A.5. Test problem #5 layout

Test Problem #5 Layout		
Part Type	Demand	Processing Alternatives (Unit Processing Times)
A	14	opr1A (9) - opr2A (6)
		opr3A (6) - opr4A (9) - opr5A (6)
		opr6A (9) - opr7A (6) - opr8A (3)
B	9	opr1B (7) - opr2B (4) - opr3B (3)
		opr4B (5) - opr5B (5) - opr6B (9)
		opr7B (10) - opr8B (5)
C	6	opr1C (7) - opr2C (6) - opr3C (5)
		opr4C (10) - opr5C (5) - opr6C (4) - opr7C (7)
D	5	opr1D (7) - opr2D (9)
		opr3D (6) - opr4D (8) - opr5D (10)
E	12	opr1E (7) - opr2E (6) - opr3E (9)
		opr4E (9) - opr5E (4) - opr6E (5)
		opr7E (10) - opr8E (7)
Tool Type (Size)	Availability	Tool's Operations
t1 (2)	2	opr1A, opr1C, opr1D
t2 (2)	1	opr2A
t3 (2)	3	opr3A, opr5A, opr7B, opr2C, opr2D, opr7E
t4 (1)	1	opr8B
t5 (2)	1	opr3C
t6 (2)	1	opr6A
t7 (1)	2	opr7A, opr5C, opr3D, opr5E, opr6E
t8 (1)	1	opr8A
t9 (1)	3	opr4A, opr1B, opr4C, opr3E, opr4E
t10 (1)	1	opr2B
t11 (2)	1	opr3B, opr6C, opr5D
t12 (1)	1	opr4B, opr4D, opr1E
t13 (1)	2	opr5B, opr7C, opr2E, opr8E
t14 (1)	1	opr6B
Machines		
Number of Machines		10
Available Processing Times		70
Slot Capacities		3

Table A.6. Test problem #6 layout

Test Problem #6 Layout		
Part Type	Demand	Processing Alternatives (Unit Processing Times)
A	14	opr1A (8) - opr2A (8)
		opr3A (10) - opr4A (7) - opr5A (10)
		opr6A (7) - opr7A (7) - opr8A (6)
B	9	opr1B (4) - opr2B (7) - opr3B (6)
		opr4B (7) - opr5B (5) - opr6B (5)
		opr7B (5) - opr8B (4)
		opr9B (10) -opr10B (9) - opr11B (10)
C	6	opr1C (8) - opr2C (7) - opr3C (8)
		opr4C (6) - opr5C (6) - opr6C (8) - opr7C (4)
		opr8C (6) - opr9C (9) - opr10C (4)
D	5	opr1D (7) - opr2D (7)
		opr3D (4) - opr4D (9) - opr5D (9)
E	12	opr1E (4) - opr2E (10) - opr3E (5)
		opr4E (8) - opr5E (4) - opr6E (9)
		opr7E (5) - opr8E (6)
		opr9E (8) - opr10E (4) - opr11E (6)
Tool Type (Size)	Availability	Tool's Operations
t1 (2)	1	opr1A, opr5C
t2 (2)	2	opr2A, opr2C, opr3E
t3 (1)	1	opr1C, opr4C, opr4E
t4 (2)	3	opr3A, opr7A, opr11B, opr6C, opr5E, opr8E
t5 (2)	2	opr4A, opr10B, opr8C
t6 (2)	2	opr9C, opr2E, opr5D
t7 (1)	2	opr5A, opr9B, opr7C
t8 (2)	1	opr8B, opr1E
t9 (1)	2	opr6A, opr7B, opr3C, opr11E
t10 (2)	1	opr6B, opr10C, opr10E
t11 (2)	1	opr5B, opr2D, opr3D
t12 (1)	1	opr8A, opr4B, opr1D
t13 (1)	1	opr4D
t14 (1)	1	opr3B
t15 (1)	1	opr1B, opr7E
t16 (1)	1	opr2B, opr6E, opr9E
Machines		
Number of Machines		9
Available Processing Times		70
Slot Capacities		4

Table A.7. Test problem #7 layout

Test Problem #7 Layout		
Part Type	Demand	Processing Alternatives (Unit Processing Times)
A	10	opr1A (6) - opr2A (8) - opr3A (7)
		opr4A (6) - opr5A (6) - opr6A (6)
		opr7A (10) - opr8A (9) - opr9A (9) - opr10A (5)
		opr11A (6) - opr12A (5)
B	8	opr1B (9) - opr2B (6) - opr3B (7)
		opr4B (5) - opr5B (5) - opr6B (9)
		opr7B (9) - opr8B (9) - opr9B (7)
C	4	opr1C (5) - opr2C (10)
		opr3C (8) - opr4C (10) - opr5C (8) - opr6C (10)
		opr7C (9) - opr8C (8)
		opr9C (10) - opr10C (5) - opr11C (7)
D	7	opr1D (9) - opr2D (8) - opr3D (9)
		opr4D (6) - opr5D (10)
E	5	opr1E (10) -opr2E (5) - opr3E (5)
		opr4E (5) - opr5E (10) - opr6E (9)
F	5	opr1F (10) - opr2F (5) - opr3F (5)
		opr4F (8) - opr5F (9) - opr6F (10) - opr7F (5)
G	9	opr1G (8) - opr2G (7) - opr3G (7)
		opr4G (7) - opr5G (9) - opr6G (9)
		opr7G (10) - opr8G (6) - opr9G (9) - opr10G (8)
Tool Type (Size)	Availability	Tool's Operations
t1 (1)	1	opr1A, opr8B, opr1G
t2 (1)	2	opr2A, opr3B, opr7B, opr1D, opr3E, opr7F
t3 (1)	2	opr3A, opr6B, opr9B, opr2E, opr6F, opr2G
t4 (1)	1	opr4A
t5 (1)	2	opr5A, opr5B, opr7C, opr4E, opr3G, opr9G
t6 (2)	1	opr6A, opr2B, opr8C, opr5E
t7 (1)	2	opr7A, opr1C, opr1E, opr6E, opr4G
t8 (1)	1	opr8A, opr1B, opr4B, opr1F
t9 (2)	2	opr9A, opr12A, opr6C, opr2D, opr2F, opr5F, opr6G, opr10G
t10 (2)	1	opr11A, opr5C, opr9C, opr5G
t11 (2)	2	opr10A, opr3C, opr11C, opr5D, opr3F, opr7G, opr8G
t12 (2)	1	opr4C, opr10C, opr3D
t13 (2)	1	opr2C, opr4D, opr4F

Table A.7 continued

Machines	
Number of Machines	9
Available Processing Times	90
Slot Capacities	3

Table A.8. Test problem #8 layout

Test Problem #8 Layout		
Part Type	Demand	Processing Alternatives (Unit Processing Times)
A	10	opr1A (10) - opr2A (5) - opr3A (8)
		opr4A (9) - opr5A (6) - opr6A (8)
		opr7A (7) - opr8A (5) - opr9A (9) - opr10A (10)
B	8	opr1B (7) - opr2B (6) - opr3B (6)
		opr4B (7) - opr5B (8) - opr6B (6)
		opr7B (6) - opr8B (10) - opr9B (6)
C	4	opr1C (6) - opr2C (7)
		opr3C (5) - opr4C (5) - opr5C (10) - opr6C (10)
		opr7C (9) - opr8C (7)
		opr9C (5) - opr10C (10) - opr11C (10)
D	7	opr1D (8) - opr2D (9) - opr3D (8)
		opr4D (10) - opr5D (8)
E	5	opr1E (6) - opr2E (5) - opr3E (5)
		opr4E (8) - opr5E (7) - opr6E (6)
F	5	opr1F (10) - opr2F (10) - opr3F (6)
		opr4F (9) - opr5F (9) - opr6F (10) - opr7F (5)
G	9	opr1G (7) - opr2G (8) - opr3G (8)
		opr4G (6) - opr5G (6) - opr6G (9)
		opr7G (5) - opr8G (9) - opr9G (10) - opr10G (9)
H	9	opr1H (6) - opr2H (10) - opr3H (6)
		opr4H (10) - opr5H (5) - opr6H (8)
I	14	opr1I (9) - opr2I (9) - opr3I (5)
		opr4I (7) - opr5I (10)
J	5	opr1J (8) - opr2J (5) - opr3J (9)
		opr4J (10) - opr5J (6)
		opr6J (8) - opr7J (7) - opr8J (9)

Table A.8 continued

Test Problem #8 Layout		
Tool Type (Size)	Availability	Tool's Operations
t1 (1)	1	opr1A, opr8B, opr1G, opr1I
t2 (2)	1	opr2A, opr3B, opr7B, opr1D, opr3E, opr7F
t3 (1)	1	opr3A, opr6B, opr9B, opr2E, opr6F, opr2G
t4 (1)	1	opr4A, opr3I
t5 (2)	1	opr5A, opr5B, opr7C, opr4E, opr3G, opr9G
t6 (1)	1	opr6A, opr2B, opr8C, opr5E
t7 (1)	1	opr7A, opr1C, opr1E, opr6E, opr4G
t8 (2)	1	opr8A, opr1B, opr4B, opr1F, opr2J
t9 (2)	1	opr9A, opr6C, opr2D, opr2F, opr5F, opr6G, opr10G, opr8J
t10 (1)	1	opr5C, opr9C, opr5G
t11 (1)	1	opr10A, opr3C, opr11C, opr5D, opr3F, opr7G, opr8G, opr2I
t12 (2)	1	opr4C, opr10C, opr3D, opr1J
t13 (1)	1	opr2C, opr4D, opr4F, opr6J
t14 (1)	1	opr5I
t15 (2)	1	opr6H, opr4I, opr5J
t16 (1)	1	opr1H, opr4H, opr3J
t17 (1)	1	opr2H, opr5H, opr7J
t18 (1)	1	opr3H, opr4J
Machines		
Number of Machines		6
Available Processing Times		240
Slot Capacities		4

Table A.9. Test problem #9 layout

Test Problem #9 Layout		
Part Type	Demand	Processing Alternatives (Unit Processing Times)
A	10	opr1A (10) - opr2A (5) - opr3A (8)
		opr4A (9) - opr5A (6) - opr6A (8)
		opr7A (7) - opr8A (5) - opr9A (9) - opr10A (10)
B	8	opr1B (7) - opr2B (6) - opr3B (6)
		opr4B (7) - opr5B (8) - opr6B (6)
		opr7B (6) - opr8B (10) - opr9B (6)
C	4	opr1C (6) - opr2C (7)
		opr3C (5) - opr4C (5) - opr5C (10) - opr6C (10)
		opr7C (9) - opr8C (7)
		opr9C (5) - opr10C (10) - opr11C (10)
D	7	opr1D (8) - opr2D (9) - opr3D (8)
		opr4D (10) - opr5D (8)
		opr6D (7) - opr7D (7)
E	5	opr1E (6) - opr2E (5) - opr3E (5)
		opr4E (8) - opr5E (7) - opr6E (6)
		opr7E (9) - opr8E (8)
F	5	opr1F (10) -opr2F (10) - opr3F (6)
		opr4F (9) - opr5F (9) - opr6F (10) - opr7F (5)
		opr8F (8) - opr9F (10) - opr10F (9)
G	9	opr1G (7) - opr2G (8) - opr3G (8)
		opr4G (6) - opr5G (6) - opr6G (9)
		opr7G (5) - opr8G (9) - opr9G (10) - opr10G (9)
		opr11G (10) - opr12G (11) - opr13G (12)
H	9	opr1H (6) - opr2H (10) - opr3H (6)
		opr4H (10) - opr5H (5) - opr6H (8)
		opr7H (10) - opr8H (10)
I	14	opr1I (9) - opr2I (9) - opr3I (5)
		opr4I (7) - opr5I (10)
		opr6I (8) - opr7I (5) - opr8I (10)
J	5	opr1J (8) - opr2J (5) - opr3J (9)
		opr4J (10) - opr5J (6)
		opr6J (8) - opr7J (7) - opr8J (9)

Table A.9 continued

Test Problem #9 Layout		
Tool Type (Size)	Availability	Tool's Operations
t1 (1)	1	opr1A, opr8B, opr1G, opr1I
t2 (1)	1	opr2A, opr3B, opr7B, opr1D, opr3E, opr7F
t3 (1)	1	opr3A, opr6B, opr9B, opr2E, opr6F, opr2G
t4 (1)	1	opr4A, opr3I
t5 (2)	1	opr5A, opr5B, opr7C, opr4E, opr3G, opr9G
t6 (1)	1	opr6A, opr2B, opr8C, opr5E
t7 (1)	1	opr7A, opr1C, opr1E, opr6E, opr4G
t8 (1)	1	opr8A, opr1B, opr4B, opr1F, opr2J, opr7E
t9 (2)	1	opr9A, opr6C, opr2D, opr2F, opr5F, opr6G, opr10G, opr8J, opr6D, opr8E
t10 (1)	1	opr5C, opr9C, opr5G, opr7D
t11 (1)	1	opr10A, opr3C, opr11C, opr5D, opr3F, opr7G, opr8G, opr2I
t12 (2)	1	opr4C, opr10C, opr3D, opr1J
t13 (1)	1	opr2C, opr4D, opr4F, opr6J
t14 (1)	1	opr5I
t15 (2)	1	opr6H, opr4I, opr5J
t16 (1)	1	opr1H, opr4H, opr3J
t17 (1)	1	opr2H, opr5H, opr7J
t18 (1)	1	opr3H, opr4J
t19 (2)	1	opr8F, opr7H, opr6I
t20 (1)	1	opr9F, opr11G, opr8H, opr7I
t21 (2)	1	opr10F, opr12G, opr8I
t22 (3)	1	opr13G
Machines		
Number of Machines		6
Available Processing Times		240
Slot Capacities		5

Table A.10. Test problem #10 layout

Test Problem #10 Layout		
Part Type	Demand	Processing Alternatives (Unit Processing Times)
A	10	opr1A (10) - opr2A (5) - opr3A (8)
		opr4A (9) - opr5A (6) - opr6A (8)
		opr7A (7) - opr8A (5) - opr9A (9) - opr10A (10)
B	8	opr1B (7) - opr2B (6) - opr3B (6)
		opr4B (7) - opr5B (8) - opr6B (6)
		opr7B (6) - opr8B (10) - opr9B (6)
C	4	opr1C (6) - opr2C (7)
		opr3C (5) - opr4C (5) - opr5C (10) - opr6C (10)
		opr7C (9) - opr8C (7)
		opr9C (5) - opr10C (10) - opr11C (10)
D	7	opr1D (8) - opr2D (9) - opr3D (8)
		opr4D (10) - opr5D (8)
		opr6D (7) - opr7D (7)
E	5	opr1E (6) - opr2E (5) - opr3E (5)
		opr4E (8) - opr5E (7) - opr6E (6)
		opr7E (9) - opr8E (8)
F	5	opr1F (10) -opr2F (10) - opr3F (6)
		opr4F (9) - opr5F (9) - opr6F (10) - opr7F (5)
		opr8F (8) - opr9F (10) - opr10F (9)
G	9	opr1G (7) - opr2G (8) - opr3G (8)
		opr4G (6) - opr5G (6) - opr6G (9)
		opr7G (5) - opr8G (9) - opr9G (10) - opr10G (9)
		opr11G (10) - opr12G (11) - opr13G (12)
H	9	opr1H (6) - opr2H (10) - opr3H (6)
		opr4H (10) - opr5H (5) - opr6H (8)
		opr7H (10) - opr8H (10)

Table A.10 continued

Test Problem #10 Layout		
Tool Type (Size)	Availability	Tool's Operations
t1 (1)	1	opr1A, opr8B, opr1G
t2 (1)	1	opr2A, opr3B, opr7B, opr1D, opr3E, opr7F
t3 (1)	1	opr3A, opr6B, opr9B, opr2E, opr6F, opr2G
t4 (1)	1	opr4A
t5 (2)	1	opr5A, opr5B, opr7C, opr4E, opr3G, opr9G
t6 (1)	1	opr6A, opr2B, opr8C, opr5E
t7 (1)	1	opr7A, opr1C, opr1E, opr6E, opr4G
t8 (1)	1	opr8A, opr1B, opr4B, opr1F, opr7E
t9 (2)	2	opr9A, opr6C, opr2D, opr2F, opr5F, opr6G, opr10G, opr6D, opr8E
t10 (1)	1	opr5C, opr9C, opr5G, opr7D
t11 (1)	1	opr10A, opr3C, opr11C, opr5D, opr3F, opr7G, opr8G
t12 (2)	1	opr4C, opr10C, opr3D
t13 (1)	1	opr2C, opr4D, opr4F
t14 (2)	1	opr13G
t15 (2)	1	opr6H
t16 (2)	1	opr1H, opr4H
t17 (2)	1	opr2H, opr5H
t18 (2)	1	opr3H
t19 (2)	1	opr8F, opr7H
t20 (3)	1	opr9F, opr11G, opr8H
t21 (2)	1	opr10F, opr12G
Machines		
Number of Machines		7
Available Processing Times		200
Slot Capacities		5

APPENDIX B: ANALYSIS PROCEDURE FOR THE DEVELOPMENT OF THE HEURISTIC ALGORITHM FOR OPERATION LOADING

Table B.1. Analysis procedure for operation loading heuristic algorithm (problem #1)

Problem #1

1st Stage ANOVA Table

Source	SS	DF	MSE	F
SS_A	156.16	4	39.04	204.51
SS_B	268.18	2	134.09	702.42
SS_C	0.24	2	0.12	0.64
SS_D	0.53	2	0.26	1.39
SS_{error}	23.67	124	0.19	
SS_t	448.79			

Average Fitness Values

B1	9.14
B2	12.15
B3	12.10

p-values of Paired t-tests

	B1
B2	0.00
B3	0.00

B1 is significantly better than B2 and B3. so B1 is fixed

2nd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_A	33.65	4	8.41	356.04
SS_C	0.05	2	0.03	1.12
SS_D	0.03	2	0.02	0.73
SS_{error}	0.85	36	0.02	
SS_t	34.58			

Average Fitness Values

A1	8.08
A2	10.36
A3	9.52
A4	8.21
A5	9.51

p-values of Paired t-tests

	A1
A2	0.00
A3	0.00
A4	0.24
A5	0.00

A1 is not significantly better than A4

Table B.2. Analysis procedure for operation loading heuristic algorithm (problem #2)

Problem #2

1st Stage ANOVA Table

Source	SS	DF	MSE	F
SS_A	1123.82	4	280.96	416.81
SS_B	1847.84	2	923.92	1370.69
SS_C	7.57	2	3.78	5.62
SS_D	0.70	2	0.35	0.52
SS_{error}	83.58	124	0.67	
SS_t	3063.52			

Average Fitness Values

B1	20.25
B2	28.18
B3	28.02

p-values of Paired t-tests

	B1
B2	0.00
B3	0.00

B1 is significantly better than B2 and B3. so B1 is fixed

2nd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_A	267.07	4	66.77	540.21
SS_C	1.08	2	0.54	4.36
SS_D	0.28	2	0.14	1.13
SS_{error}	4.45	36	0.12	
SS_t	272.87			

Average Fitness Values

A1	16.12
A2	22.90
A3	20.42
A4	22.44
A5	19.36

p-values of Paired t-tests

	A1
A2	0.00
A3	0.00
A4	0.00
A5	0.00

A1 is significantly better than A2. A3. A4 and A5. so A1 is fixed

3rd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_C	0.07	2	0.03	4.32
SS_D	0.03	2	0.02	1.89
SS_{error}	0.03	4	0.01	
SS_t	0.13			

No significant factor since $F_{2.4.0.95} = 6.94$

Table B.3. Analysis procedure for operation loading heuristic algorithm (problem #3)

Problem #3

1st Stage ANOVA Table

Source	SS	DF	MSE	F
SS_A	234.48	4	58.62	199.22
SS_B	105.61	2	52.81	179.46
SS_C	0.12	2	0.06	0.21
SS_D	0.25	2	0.13	0.43
SS_{error}	36.49	124	0.29	
SS_t	376.96			

Average Fitness Values

A1	15.78
A2	19.90
A3	17.53
A4	17.39
A5	17.53

p-values of Paired t-tests

	A1
A2	0.00
A3	0.00
A4	0.00
A5	0.00

A1 is significantly better than A2. A3. A4 and A5. so A1 is fixed

2nd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_B	32.45	2	16.23	63.55
SS_C	0.53	2	0.27	1.04
SS_D	0.03	2	0.01	0.05
SS_{error}	5.11	20	0.26	
SS_t	38.12			

Average Fitness Values

B1	14.23
B2	16.50
B3	16.60

p-values of Paired t-tests

	B1
B2	0.00
B3	0.00

B1 is significantly better than B2 and B3. so B1 is fixed

3rd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_C	0.04	2	0.02	0.62
SS_D	0.06	2	0.03	0.99
SS_{error}	0.12	4	0.03	
SS_t	0.22			

No significant factor since $F_{2.4.0.95} = 6.94$

Table B.4. Analysis procedure for operation loading heuristic algorithm (problem #4)

Problem #4

1st Stage ANOVA Table

Source	SS	DF	MSE	F
SS_A	716.39	4	179.10	95.53
SS_B	617.49	2	308.75	164.68
SS_C	8.07	2	4.03	2.15
SS_D	5.47	2	2.74	1.46
SS_{error}	232.48	124	1.87	
SS_t	1579.90			

Average Fitness Values

B1	23.88
B2	28.94
B3	27.59

p-values of Paired t-tests

	B1
B2	0.00
B3	0.00

B1 is significantly better than B2 and B3. so B1 is fixed

2nd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_A	365.47	4	91.37	296.13
SS_C	5.44	2	2.72	8.82
SS_D	5.88	2	2.94	9.54
SS_{error}	11.11	36	0.31	
SS_t	387.91			

Average Fitness Values

A1	19.55
A2	28.49
A3	23.39
A4	24.44
A5	23.54

p-values of Paired t-tests

	A1
A2	0.00
A3	0.00
A4	0.00
A5	0.00

A1 is significantly better than A2. A3. A4 and A5. so A1 is fixed

3rd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_C	0.04	2	0.02	0.33
SS_D	0.98	2	0.49	8.45
SS_{error}	0.23	4	0.06	
SS_t	1.26			

Table B.4 continued

Problem #4

Average Fitness Values	
D1	19.64
D2	19.11
D3	19.91

p-values of Paired t-tests

	D2
D1	0.02
D3	0.01

D2 is significantly better than D1 and D3

Table B.5. Analysis procedure for operation loading heuristic algorithm (problem #5)

Problem #5

1st Stage ANOVA Table

Source	SS	DF	MSE	F
SS_A	1290.98	4	322.74	528.16
SS_B	638.69	2	319.35	522.60
SS_C	18.06	2	9.03	14.78
SS_D	8.86	2	4.43	7.25
SS_{error}	75.77	124	0.61	
SS_t	2032.37			

Average Fitness Values

A1	15.68
A2	24.49
A3	17.91
A4	16.99
A5	20.20

p-values of Paired t-tests

	A1
A2	0.00
A3	0.00
A4	0.00
A5	0.00

A1 is significantly better than A2. A3. A4 and A5. so A1 is fixed

2nd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_B	126.24	2	63.12	462.22
SS_C	2.33	2	1.16	8.52
SS_D	2.55	2	1.27	9.33
SS_{error}	2.73	20	0.14	
SS_t	133.85			

Average Fitness Values

B1	12.73
B2	17.86
B3	16.43

p-values of Paired t-tests

	B1
B2	0.00
B3	0.00

B1 is significantly better than B2 and B3. so B1 is fixed

Table B.5 continued

Problem #5

3rd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_C	1.28	2	0.64	7.25
SS_D	0.95	2	0.47	5.37
SS_{error}	0.35	4	0.09	
SS_t	2.58			

Average Fitness Values

C1	12.27
C2	12.99
C3	12.93

p-values of Paired t-tests

	C1
C2	0.00
C3	0.01

C2 is significantly better than C1 and C3

Table B.6. Analysis procedure for operation loading heuristic algorithm (problem #6)

Problem #6

1st Stage ANOVA Table

Source	SS	DF	MSE	F
SS_A	3322.29	4	830.57	407.38
SS_B	2260.99	2	1130.49	554.48
SS_C	55.86	2	27.93	13.70
SS_D	40.77	2	20.38	10.00
SS_{error}	252.82	124	2.04	
SS_t	5932.73			

Average Fitness Values

B1	23.93
B2	32.09
B3	33.06

p-values of Paired t-tests

	B1
B2	0.00
B3	0.00

B1 is significantly better than B2 and B3. so B1 is fixed

Table B.6 continued

Problem #6

2nd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_A	1046.51	4	261.63	459.38
SS_C	17.54	2	8.77	15.40
SS_D	18.76	2	9.38	16.47
SS_{error}	20.50	36	0.57	
SS_t	1103.30			

Average Fitness Values

A1	16.44
A2	30.22
A3	21.32
A4	27.65
A5	24.04

p-values of Paired t-tests

	A1
A2	0.00
A3	0.00
A4	0.00
A5	0.00

A1 is significantly better than A2. A3. A4 and A5. so A1 is fixed

3rd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_C	3.28	2	1.64	21.44
SS_D	4.24	2	2.12	27.70
SS_{error}	0.31	4	0.08	
SS_t	7.82			

Average Fitness Values

D1	15.88
D2	17.41
D3	16.04

p-values of Paired t-tests

	D1
D2	0.00
D3	0.51

Average Fitness Values

C1	17.09
C2	15.64
C3	16.60

p-values of Paired t-tests

	C2
C1	0.00
C3	0.00

C2 is significantly better than C1 and C3

D1 is not significantly better than D3

Table B.7. Analysis procedure for operation loading heuristic algorithm (problem #7)

Problem #7

1st Stage ANOVA Table

Source	SS	DF	MSE	F
SS_A	1503.56	4	375.89	329.64
SS_B	1364.05	2	682.02	598.10
SS_C	9.48	2	4.74	4.16
SS_D	30.19	2	15.09	13.24
SS_{error}	141.40	124	1.14	
SS_t	3048.68			

Average Fitness Values

B1	17.77
B2	24.39
B3	24.63

p-values of Paired t-tests

	B1
B2	0.00
B3	0.00

B1 is significantly better than B2 and B3. so B1 is fixed

2nd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_A	245.07	4	61.27	485.84
SS_C	1.27	2	0.63	5.02
SS_D	17.69	2	8.85	70.16
SS_{error}	4.54	36	0.13	
SS_t	268.57			

Average Fitness Values

A1	14.69
A2	21.87
A3	17.00
A4	17.27
A5	18.01

p-values of Paired t-tests

	A1
A2	0.00
A3	0.00
A4	0.00
A5	0.00

A1 is significantly better than A2. A3. A4 and A5. so A1 is fixed

Table B.7 continued

Problem #7

3rd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_C	0.11	2	0.06	0.47
SS_D	3.28	2	1.64	13.89
SS_{error}	0.47	4	0.12	
SS_t	3.86			

Average Fitness Values

D1	13.87
D2	14.88
D3	15.31

p-values of Paired t-tests

	D1
D2	0.01
D3	0.00

D1 is significantly better than D2 and D3

Table B.8. Analysis procedure for operation loading heuristic algorithm (problem #8)

Problem #8

1st Stage ANOVA Table

Source	SS	DF	MSE	F
SS_A	12327.87	4	3081.97	1915.19
SS_B	1087.11	2	543.56	337.78
SS_C	7.61	2	3.80	2.36
SS_D	6.80	2	3.40	2.11
SS_{error}	199.54	124	1.61	
SS_t	13628.94			

Average Fitness Values

A1	42.62
A2	72.63
A3	56.49
A4	59.40
A5	56.23

p-values of Paired t-tests

	A1
A2	0.00
A3	0.00
A4	0.00
A5	0.00

A1 is significantly better than A2. A3. A4 and A5. so A1 is fixed

Table B.8 continued

Problem #8

2nd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_B	138.24	2	69.12	129.56
SS_C	3.90	2	1.95	3.66
SS_D	0.07	2	0.04	0.07
SS_{error}	10.67	20	0.53	
SS_t	152.89			

Average Fitness Values

B1	39.42
B2	44.32
B3	44.11

p-values of Paired t-tests

	B1
B2	0.00
B3	0.00

B1 is significantly better than B2 and B3. so B1 is fixed

3rd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_C	1.16	2	0.58	174.68
SS_D	0.04	2	0.02	5.53
SS_{error}	0.01	4	0.00	
SS_t	1.21			

Average Fitness Values

C1	39.92
C2	39.12
C3	39.21

p-values of Paired t-tests

	C2
C1	0.04
C3	0.76

C2 is not significantly better than C3

Table B.9. Analysis procedure for operation loading heuristic algorithm (problem #9)

Problem #9

1st Stage ANOVA Table

Source	SS	DF	MSE	F
SS_A	11202.30	4	2800.58	1171.37
SS_B	1764.27	2	882.13	368.96
SS_C	0.63	2	0.32	0.13
SS_D	6.61	2	3.30	1.38
SS_{error}	296.47	124	2.39	
SS_t	13270.27			

Average Fitness Values

A1	33.15
A2	61.72
A3	44.99
A4	46.87
A5	44.86

p-values of Paired t-tests

	A1
A2	0.00
A3	0.00
A4	0.00
A5	0.00

A1 is significantly better than A2. A3. A4 and A5. so A1 is fixed

2nd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_B	158.79	2	79.39	263.22
SS_C	0.18	2	0.09	0.30
SS_D	0.44	2	0.22	0.73
SS_{error}	6.03	20	0.30	
SS_t	165.44			

Average Fitness Values

B1	29.72
B2	34.88
B3	34.84

p-values of Paired t-tests

	B1
B2	0.00
B3	0.00

B1 is significantly better than B2 and B3. so B1 is fixed

3rd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_C	0.37	2	0.19	3.51
SS_D	0.01	2	0.01	0.10
SS_{error}	0.21	4	0.05	
SS_t	0.60			

No significant factor since $F_{2.4.0.95} = 6.94$

Table B.10. Analysis procedure for operation loading heuristic algorithm (problem #10)

Problem #10

1st Stage ANOVA Table

Source	SS	DF	MSE	F
SS_A	9851.83	4	2462.96	959.44
SS_B	365.82	2	182.91	71.25
SS_C	8.69	2	4.35	1.69
SS_D	2.35	2	1.18	0.46
SS_{error}	318.32	124	2.57	
SS_t	10547.01			

Average Fitness Values

A1	30.11
A2	56.56
A3	40.00
A4	43.66
A5	39.80

p-values of Paired t-tests

	A1
A2	0.00
A3	0.00
A4	0.00
A5	0.00

A1 is significantly better than A2. A3. A4 and A5. so A1 is fixed

2nd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_B	202.21	2	101.11	225.46
SS_C	1.02	2	0.51	1.14
SS_D	0.21	2	0.10	0.23
SS_{error}	8.97	20	0.45	
SS_t	212.41			

Average Fitness Values

B1	26.25
B2	32.31
B3	31.76

p-values of Paired t-tests

	B1
B2	0.00
B3	0.00

B1 is significantly better than B2 and B3. so B1 is fixed

3rd Stage ANOVA Table

Source	SS	DF	MSE	F
SS_C	0.23	2	0.12	1.19
SS_D	0.64	2	0.32	3.26
SS_{error}	0.39	4	0.10	
SS_t	1.27			

No significant factor since $F_{2.4.0.95} = 6.94$

REFERENCES

Albey, E., 2006, *A Hierarchical Approach for Responsive Production Planning and Control System in Flexible Manufacturing Systems*, Unpublished M.S. Thesis, Boğaziçi University.

Arikan, M. and Erol, S., 2006, "Meta-heuristic Approaches for Part Selection and Tool Allocation in Flexible Manufacturing Systems", *International Journal of Computer Integrated Manufacturing*, 19 (4), 315–325.

Browne, J., Dubois, D., Rathmill, K. Sethi, S. and Stecke, K., 1984, "Classification of Flexible Manufacturing Systems", *FMS Magazine*, 2, No.3, 114–117.

Carter, M. F., 1986, "Designing flexibility into automated manufacturing systems", *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems: Operation Research Models and Applications (K. E. Stecke and R. Suri, eds.),* Elsevier Science Publishers B. V., New York, pp. 107–118.

Goldberg, D. E., 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley Longman, Inc.

Grieco, A., Semeraro, Q. and Tolio, T., 2001, "A Review of Different Approaches to the FMS Loading Problem", *The International Journal of Flexible Manufacturing Systems*, 13, 361–384.

Kumar, N. and Shanker, K., 2000, "A genetic algorithm for FMS part type selection and machine loading", *International Journal of Production Research*, 38, 3861–3887.

Kumar, N. and Shanker, K., 2001, "Comparing the Effectiveness of Workload Balancing Objectives in FMS Loading", *International Journal of Production Research*, 39(5), 843–871.

Reeves, C. R. and Rowe, J. E., 2003, *Genetic Algorithms – Principles and Perspectives, A Guide to GA Theory*, Kluwer Academic Publishers.

Sarma, U.M.B.S., Kant, S., Rai, R. and Tiwari, M. K., 2002, "Modeling the machine loading problem of FMSs and its solution using a tabu-search-based heuristic", *International Journal of Computer Integrated Manufacturing*, 15(4), 285–295.

Singh, N., 1996, *Systems Approach to Computer-Integrated Design and Manufacturing*, Wiley.

Stecke, K. E., 1983, "Formulation of solution of non integer production planning problems in flexible manufacturing systems", *Management Science*, 29, 273–288.

Stecke, K. E., 1985, "Design, planning, scheduling and control problems of flexible manufacturing systems", *Annals of Operations Research*, 3, 3–12.

Stecke, K. E. and Solberg, J. J., 1981, "Loading and control policies for a flexible manufacturing system", *International Journal of Production Research*, 19 (5), 481–490.

Swarnkar, R. and Tiwari, M.K., 2004, Modeling machine loading problem of FMSs and its solution methodology using a hybrid tabu search and simulated annealing-based heuristic approach. Robotics and Computer-Integrated Manufacturing 20, 199–209.

Shanker, K. and Srinivasulu, A., 1989, "Some solution methodologies for a loading problem in a flexible manufacturing system", *International Journal of Production Research*, 27 (6), 1019–1034.

Shanker, K. and Tzen, Y.-J. J., 1985, "A loading and dispatching problem in a random FMS", *International Journal of Production Research*, 23 (3), 579–593.

Martello, S. and Paolo, T., 1990, *Knapsack Problems: Algorithms and Computer Implementations*, J. Wiley & Sons.

Mukhopadhyay, S. K., Midha, S. and Muhlikrishna, 1992, "A heuristic procedure for loading problems in FMS", *International Journal of Production Research*, 27 (6), 1019–1034.

Mukhopadhyay, S. K., Singh, M. K. and Srivastava, R., 1998, "FMS machine loading: a simulated annealing approach", *International Journal of Production Research*, 36 (6), 1529–1547.

Tiwari, M. K., Hazarika, B., Vidyarthi, N. K., Jaggi, P. and Mukhopadhyay, S. K., 1997, "A heuristic solution to machine loading problem of a FMS and its Petri net model", *International Journal of Production Research*, 35(8), 2269–2284.

Tiwari, M. K. and Vidyarthi, N. K., 2000, "Solving machine loading problem in flexible manufacturing system using genetic algorithm based heuristic approach", *International Journal of Production Research*, 38, 3357–3384.

Vidyarthi, N. K. and Tiwari, M. K., 2001, "Machine loading problem of FMS: a fuzzy-based heuristic approach", *International Journal of Production Research*, 39(5), 953–979.