INT. J. PROD. RES., 2000, VOL. 38, NO. 14, 3357–3384

# Solving machine loading problems in a flexible manufacturing system using a genetic algorithm based heuristic approach

M. K. TIWARI†* and N. K. VIDYARTHI‡

The machine-loading problem of a flexible manufacturing system (FMS) has been recognized as one of the most important planning problems. In this research, a Genetic Algorithm (GA) based heuristic is proposed to solve the machine loading problem of a random type FMS. The objective of the loading problems is to minimize the system unbalance and maximize the throughput, satisfying the technological constraints such as availability of machining time, and tool slots. The proposed GA-based heuristic determines the part type sequence and the operation-machine allocation that guarantee the optimal solution to the problem, rather than using fixed predetermined part sequencing rules. The efficiency of the proposed heuristic has been tested on ten sample problems and the results obtained have been compared with those of existing methods.



## 1. Introduction

Many firms have adopted flexible manufacturing systems (FMSs) as a means to produce high quality products with small lead times in order to meet the growing requirements of customized production. The development of FMS is mainly concerned with achieving efficiency in a well-balanced transfer line, while retaining the flexibility of a low volume job-shop-type conventional manufacturing system. There are a variety of problems to be addressed for the successful development and implementation of an FMS. In general, FMS operational decisions consist of pre-release and post-release decisions. FMS planning problems — also known as pre-release decisions — take into account the pre-arrangement of parts and tools before the process of FMS begins. FMS scheduling problems, which come under the category of post-release decisions, deal with the sequencing and routing of the parts when the system is in operation. Post-release decisions problems are:

(a) part type selection;
(b) machine grouping;
(c) determination of production ratio;
(d) batching of the part types;
(e) allocation of pallets and fixtures;
(f) allocation of operations and tools among machines (loading problem) (Stecke 1983, 1985, Sarin and Chen 1987).

* Department of Manufacturing Engineering, National Institute of Foundry and Forge Technology (NIFFT), Hatia, Ranchi – 834 003, India.
‡ Department of Mechanical Engineering, North Eastern Regional Institute of Science and Technology (NERIST), Itanagar – 791 109, India.
*To whom correspondence should be addressed. e-mail: mktog@hotmail.com

*International Journal of Production Research* ISSN 0020–7543 print/ISSN 1366–588X online 
http://www.tandf.co.uk/journals

Various methodologies and approaches have been suggested to address these problems (Buzacott and Yao 1980). Loading problems in manufacturing deal with the assignment of various resources (machines, tools, fixtures, pallets etc) to the operations of different part types that are already planned for production in a given planning horizon. Decisions pertaining to loading problems have been viewed as tactical level planning decisions that receive their inputs from the preceding decision levels; namely, grouping of resources, selection of part mixes, aggregate planning that generates the inputs to the succeeding decisions of scheduling resources, and dynamic operations planning and control. Therefore, it can be construed that loading decisions are acting as an important link between strategic and operation level decisions in manufacturing. Van Loovern *et al.* (1986), Kusiak (1985), Singhal (1978), Whitney and Gaul (1984) have discussed the interrelationships of various decisions and their hierarchies in a flexible manufacturing environment. Liang and Dutta (1993) addressed the part selection and machine loading problem simultaneously, which had been treated separately by previous researchers. A number of researchers have addressed the part selection problem using various methods (Kusiak 1984, 1985, Stecke 1985, Hwang and Shogun 1989), whereas formulations and solution methodologies for various scenarios and combinations of parameters relating to the loading problem in an FMS have atttracted the attention of numerous researchers including Liang and Dutta (1992), Tiwari *et al.* (1997), Laskari *et al.* (1987), Mukhopadhyay *et al.* (1992), Chen and Askin (1990), Shanker and Tzen (1985), Shanker and Srinivasulu (1989), Ram *et al.* (1990) etc. Part selection, machine loading, and tool configurations are three different but interlinked problems that are connected by common restrictions such as tool magazine capacity, job-tool-machine compatibility and available machine time. However, for the clarity of the problems, most researchers have treated part selection, machine loading, and tool configuration in a discrete manner.

The machine loading problem can be described as '... given a set of part types to be produced, set of tools that are needed for processing the parts on a set of machines, and using a set of resources such as material handling systems, pallets and fixtures, how the parts be assigned and tools should be allocated so that some measures of productivity is optimized'. Stecke (1983) described six objectives of the machine loading problem in a FMS:

(1) balancing the machine processing time;
(2) minimizing the number of movements;
(3) balancing the workload per machine for a system of groups of pooled machines of equal sizes;
(4) unbalancing the workload per machine for a system of groups of pooled machines of unequal sizes;
(5) filling the tool magazines as densely as possible;
(6) maximizing the sum of operations priorities.

It has been observed that, in various situations, some of the objectives are contradictory whereas in others, several of the objectives may be equally applicable. Besnet and Mize (1994) have broadly classified the approaches of solving the machine loading problem as:

(1) mathematical programming (Stecke 1993, Laskari *et al.* 1987, Shanker and Srinivasulu 1989, etc.);

(2) multicriteria decision making (Kumar *et al.* 1990, Chen and Askin 1990, etc.);
(3) heuristic oriented (Stecke and Solberg 1981, Moreno and Ding 1993, Mukhopadhyay *et al.* 1992);
(4) simulation based (Jain *et al.* 1989).

Ammons *et al.* (1985) described the bi-criterion objective for the machine-loading problem, i.e. balancing workloads and minimizing workstation visits. Shanker and Tzen (1985) also described a bi-criterion objective for a loading problem that includes balancing workloads and meeting due dates of part types. A mixed-integer programming model for the loading problem was presented along with the two heuristic procedures. They suggested that a mathematical programming approach is impractical due to large computational time requirements even for a moderate size test problem. Rajagopalan (1986) combines the loading problem with others that are found in the planning stage, such as part type selection and production ratio determination so as to achieve better production schedules by avoiding an iterative process. Mukhopadhyay *et al.* (1992), Tiwari *et al.* (1997) attempted the machine loading problem using heuristic procedures with an objective to minimize the system unbalance and maximize throughput.

Compared with other objective functions of machine loading problems – such as maximizing the work load balance on the machines, due date satisfaction, maximizing system utilization, minimizing in-process inventory, maximizing production rate, minimizing set-up and tool changing costs, minimizing flow time etc. — the one followed in this research is the minimization of system unbalance and the maximization of throughput because:

(a) owing to high capital investment in installing an FMS, it is naturally expected by management to achieve high machine utilization. This can be easily achieved by taking the above objective function, as it is concerned with the minimization of system idle time;
(b) FMSs are usually employed for medium production volume and medium part variety within a given time frame and varying processing requirements in the presence of limited resources. The most realizable objective of pursuing any loading policy will definitely be concerned with enhancing total system output, which is referred to here as throughput;
(c) according to the finding of Kim and Yano (1997), it is believed that throughput maximization by balancing the workloads on the machines often results in limiting the tardiness.

Most of the aforesaid researchers have adopted predetermined sequencing rules for the part input in order to simplify the complexities of the machine-loading problem. Shortest processing time (SPT) has been preferred, due to its ability to work better on average, as far as achieving the objectives of loading problems are concerned. In order fully to exploit the effectiveness of the heuristic proposed by Tiwari *et al.* (1997) and to ensure that optimal or near-optimal solutions are obtained in each case, it is desirable to determine the sequence of part types for practical problems rather than adopting fixed predetermined sequencing rules. In fact, simultaneous determination of a part type sequence, system unbalance and throughput, by satisfying the technological constraints (limited tool slots, machine time, etc.) while also addressing the machine-loading problem is an NP-hard problem. In this research, an

attempt has been made to solve a moderate size machine-loading problem pertaining to random FMS using a genetic-algorithm based heuristic.

It is abundantly clear from the aforementioned literature that neither the heuristic techniques nor the mathematical programming based methodologies are able to produce optimal or near-optimal solutions to the machine loading problem that can be used as inputs to tackle the scheduling and control problems of the FMS. Taking into account the limitations and inefficacies of the several heuristic procedures suggested so far to address the machine-loading problem of a random FMS, an attempt has been made here to adopt such a heuristic procedure. Such a procedure can prove to be effective in minimizing large search spaces even in the presence of difficulties such as high dimensionality, multi-modality, discontinuity, and noise, to obtain optimal or sub-optimal solution to the variety of functions. This can be achieved by a GA-based heuristic in which stochastic processes generate an initial population of sequences and the principle of natural selection/survival-of-the-fittest is applied to improve the part sequences by evaluating the performance measures, such as system unbalance and throughput using a heuristic approach. The main advantage of a GA-based heuristic lies in its ability to jump randomly from sequence to sequence allowing the solution to escape from the local optima in which other algorithms often get trapped.

While pursuing this research, we come to understand that most of the traditional methods of optimization do not perform well when they are employed in problems where the practical search space is too large. Alternatively, it can be viewed that traditional optimization methods (calculus based, integer programming, dynamic programming etc) are structurally rigid. Goldberg (1989, p. 7) pointed out the merit of GAs over traditional optimization methods in the following ways:

(a) GAs work with a coding of the parameter set, not the parameters themselves;
(b) GAs search from a population of points, not from a single point;
(c) GAs use objective function information, not the derivatives or other auxiliary knowledge;
(d) GAs use probabilistic transition rules, not deterministic rules.

The above facts about the advantages of GAs further motivated the authors to pursue this research, in which a GA-based heuristic is proposed to solve the machine loading problem of FMS. This heuristic has been tested on ten moderate-size test problems and the results have been compared with the solutions obtained by applying the heuristics reported by Tiwari *et al.* (1997), Shanker and Srinivasulu (1989), Mukhopadhyay *et al.* (1992). This research also covers the comparative study of different predetermined part type sequencing rules for solving the machine-loading problem with the proposed GA-based heuristic.

Compared with the aforementioned heuristic methods, the proposed GA-based heuristic performed quite well and superseded the other methods on these similar problems under the same assumptions. The test results for the similar problems confirm the usefulness of the proposed heuristic in providing an optimal solution to the loading problems with the objective functions of minimizing the system unbalance and maximizing throughput.

In the following, we first describe, in section 2, the background of the genetic algorithm and the structure of a simple GA. Section 3 covers the problem description for a random FMS along with the formulation of objective functions

by taking into account the technological constraints. A GA-based heuristic solution is discussed in section 4. Section 5 includes an illustrative example to demonstrate the application of the proposed methodologies. Results and discussions are given in section 6, and conclusions along with the future scope for research are detailed in section 7.

## 2. Background of the Genetic Algorithm

A Genetic Algorithm is an 'intelligent' probabilistic search algorithm that simulates the process of evolution by taking a population of solutions and applying genetic operators in each reproduction. Each solution in the population is evaluated according to some fitness measure. Fitter solutions in the population are used for reproduction. New 'offspring' solutions are generated and unfit solutions in the population are replaced. The cycle of evaluation-selection-reproduction is continued until a satisfactory solution is found (Goldberg 1989, Michalewicz 1992). Holland (1975) introduced genetic algorithms which, later on, were applied to a wide variety of problems. Some of the typical applications of GAs are as follows:

(a) travelling salesman problem (Grefenstette *et al.* 1985);
(b) Scheduling problem (Davis 1985a, Clevland and Smith 1989, Chen *et al.* 1995, Jain and Elmaraghy 1997, Lee *et al.* 1997, Herrmann *et al.* 1995, Herrmann and Lee 1995);
(c) VLSI Circuit layout design problem (Fourmann 1985);
(d) Computer aided gas pipeline operation problem (Goldberg 1987a, 1987b);
(e) communication network control problem (Cox *et al.* 1991);
(f) real time control problem in manufacturing systems (Grefenstette 1989, Lee *et al.* 1997);
(g) cellular manufacturing Gupta *et al.* 1996);
(h) pattern classification (Bandyopadhyay *et al.* 1995, Bandyopadhyay and Pal 1998).

The following brief outlines of GAs illustrate its functioning.

Holland (1975) first described a GA, which is commonly called the Simple Genetic Algorithm (SGA). The working of the SGA can best be understood by the following steps.

*Step* 1. Generate the initial population. Determine the size of the population and the maximum number of the generation.
*Step* 2. Calculate the fitness value of each member of the initial population.
*Step* 3. Calculate the selection probability of each member of the initial population using the ratio of fitness value of that initial population to the summation of the fitness values of the individual solutions.
*Step* 4. Select a pair of members (parents) that can be used for reproduction using selection probability.
*Step* 5. Apply the genetic operators such as crossover, mutation, and inversion to the parents. Replace the parents with the new offspring to form a new population. Check the size of the new population. If it is equal to the initial population size, then go to step 6, otherwise go to step 4.
*Step* 6. If the current generation is equal to the maximum number of the generation then stop, else move to step 2.

According to the aforementioned outline, the following points should be taken into account while applying the GA to any problem.

(1) representation of structure;
(2) initial population;
(3) population size;
(4) selection probability;
(5) genetic operators;
(6) termination.

A detailed description of these fundamental elements of the genetic algorithm can best be understood by consulting in section 4 and the illustrative example. After searching a large amount of the literature in the area of GA application, it has been found that there are a plethora of articles addressing the scheduling problems of FMSs, whereas its application to the machined loading problem has received very little attention so far. This research intends to demonstrate the advantage of GA applications in the area of the machine-loading problem of a random FMS that is known for its computational complexity (even for moderate size FMS).



## 3. Problem description

We have considered the loading problem for the analysis of a random FMS consisting of a number of machines. Each machine has fixed number of tool slots. Part types are arriving randomly in a given planning period and their operation times and tool slot requirements are well known. This type of problem has already been addressed by several researchers in the recent past (Shanker and Tzen 1985, Mukhopadhyay *et al.* 1992, Tiwari *et al.* 1997). The random FMS considered here is capable of performing operations that may be essential or optional. Essential operations are those that can be carried out on a particular machine using particular tool slots. Therefore, the FMS under consideration is able to delineate the flexibility pertaining to machine selection, operation processing, part sequence selection etc. It is known that the machine loading problem deals with selecting a subset of part types from a set of part types and assigning their operations to the appropriate machines in a given planning horizon in order to achieve certain performance measures of the system by taking into account the technological constraints of the system. System unbalance and throughput are the most commonly used performance measures in the context of a loading problem, whereas the common technological constraints encountered are the availability of machining time and tool slots on machines. In order to understand the complexities of a loading problem of a random FMS, consider an example that consists of four machines, each having five tool slots, and the processing time for carrying out the different operations of the part types are known. Each part type consists of four operations, which can be performed on any of the machines but the sequence of the operations remains unaltered. Different operations of the part types can be performed on different machines with unequal machining time and different tool slots. The versatility of each machine and its capability of performing many different operations enable several operation assignments to be duplicated to generate alternative part routes. Due to the above facts, it is found that there exist a fairly large number of combinations in which operations of the part type can be assigned on the different machines while satisfying the system constraints. The problem becomes more complex if some other flexibilities—such as tooling flexibilities, part movement flexibilities, etc.—are

to be considered along with the constraints of the system configuration and operational feasibility. These operation-machine allocation combinations are to be evaluated using two common yardsticks: system unbalance and throughput. System unbalance can be defined as the sum of unutilized or overutilized time on all the machines available in the system. Minimization of system unbalance is same as maximization of machine utilization, whereas 'throughput' refers to the units of part types produced.

It is very difficult to evaluate all the possible combinations of operation-machine allocation in order to achieve minimum system unbalance and maximum throughput. This is because it takes a large search space as well as significant computational time. Earlier researchers have addressed the problems by developing predetermined part-sequence-based heuristic solutions that do not guarantee optimal or near optimal solution. Therefore, the application of a local search heuristic, such as a genetic algorithm, simulated annealing algorithm, and tabu search, are to be employed to address the above problem. In this paper, we have applied GA-based heuristics to obtain the optimal or near-optimal combination of operation–machine allocations of given part types, with tool slots and machine time availability as constraints. This problem has been addressed considering the following objective functions:

(1) minimization of system unbalance alone;
(2) maximization of throughput alone;
(3) multiple objectives, i.e. minimization of system unbalance and maximization of throughput.

The following assumptions are made to minimize the complexities in analysing the loading problem of the system.

(1) All the part types and machines are simultaneously available initially.
(2) The processing time required to complete an operation that is required for an entire part type order is known.
(3) A part type that undertaken for processing is to be completed, for all its operation, before a new part type is considered. This is called non-splitting of the part type.
(4) An operation of a part type, once started on a machine, is continued until it is completed.
(5) The transportation time required to move a part type between machines is negligible.
(6) Sharing and duplication of tool slots have been ignored.

## 4. GA-based solution methodology for machine loading problem

In this section, the objective is to define the terminologies and to discuss the various design issues (e.g. representation, initialization, evaluation function, crossover, mutation, and elitist strategies etc) related to a genetic algorithm with reference to its application in solving the machine loading problems of a FMS. A genetic algorithm starts with an initial population of individuals (also know as chromosomes or strings) representing different possible solutions to a problem. The population is maintained by the iterations of the algorithm, termed as the *generation*. The fitness of each individual is evaluated in each generation and the individual is stochastically selected for a next generation based upon its fitness. New individuals, commonly known as offsprings, are produced by two genetic operators: crossover and muta-

tion. It is assumed that the offsprings inherit the good attributes from their parents in order that the average quality of the solution becomes better than that in the previous population. This evolutionary process is terminated when some specified stopping criteria are met. The major plus point of a GA is the flexibility to adopt itself to changing optimization criteria and constraints. Factors such as representation of individuals, decoding methods, initial population, the selection scheme, and the choice of genetic operators, have a tremendous influence on the performance of the genetic algorithm. In the next subsections, these factors are discussed in detail.

### 4.1. *Representation*

Representation (encoding) plays a major role in the development of GAs. Having a good representation scheme, which can describe the problem-specific characteristic well is crucial because it significantly influences all the subsequent steps of the GA. The bits (genes) could be binary, real integer number or a combination of characters. Grefenstette *et al.* (1986) discussed a representation scheme known as adjacency representation. Similarly, Goldberg and Lingle (1985) presented a permutation representation scheme. Sawaka *et al.* (1996) discussed a matrix-based representation scheme while solving the problems of flexible scheduling in a machining centre. All the above representation schemes have been examined in this problem, and we have decided to use the sequence-oriented representation scheme. For example, eight part types which are to be loaded on the machines are represented as **[**5 7 1 8 4 3 6**]**, where the numbers in the parenthesis are part types in the given sequence.



### 4.2. *Initialization*

A genetic algorithm operates on a population of individual strings. Several researchers have used either heuristic procedures or random techniques to generate feasible strings that form the initial population. The performance of the GA is better with a random start than from a preselected starting population (Anderson and Ferris 1994). It is believed that a more diverse population initiates a more effective search. It is also known that improving the average fitness value of the initial population will minimize the computation time. Therefore, in generating all the initial populations with different procedures, it is aiming to improve effectively the average fitness value of the population that does not significantly affect the diversity of the population. This is the main reason for randomly generating an initial population in this research.

### 4.3. *Evaluation of fitness function*

During each generation, chromosomes are evaluated using some measures of fitness. In most optimization applications, fitness is calculated based on the original objective function. In this research, the following objective functions have been proposed to meet the requirements of machine loading problem.

(1) *Minimize the system unbalance.* It was mentioned earlier that system unbalance is the summation of idle time or the overutilized time on all machines in the system. Say, for example, if the planning period is defined to be 8 hours (= 480 minutes) and there are four machines, the value of system unbalance will be $(4 \times 480 \text{ minutes}) = 1920$ minutes when there are no part types present in the system for processing. As a second example of system unbalance, assume that only one part type is selected and it requires

360 minutes of processing on machine 1 and 408 minutes of processing on machine 4, then the system unbalance = $(480 \times 4 - 360 - 408) = 1152$ minutes. When the objective is to minimize system unbalance, part types are arranged in a sequence and machine–operation allocations are carried out so as to maximize the machine utilization without considering the rate of increase in corresponding throughput.

For evaluating the sequence with the objective of minimizing system unbalance by satisfying the technological constraints such as machining time and tool slots, the fitness function can be expressed as:

$$\text{Maximize } f_1 = \frac{\text{SU}_{\max} - \text{SU}_{\text{seq}}}{\text{SU}_{\max} - \text{SU}_{\min}}, \tag{1}$$

where

$\text{SU}_{\max}$ is the maximum system unbalance (= 1920 min),
$\text{SU}_{\min}$ is the minimum system unbalance (= 0 min),
$\text{SU}_{\text{Seq}}$ is the system unbalance corresponding to a particular part type of sequence.

(2) *Maximize throughput*. Throughput is the units of part types produced from the system without violating the system constraints. When the objective is to maximize throughput, part types are arranged in a sequence and machine-operation allocations are carried out to ensure the maximization of the number of part types that can be produced that satisfy the system constraints, without considering the rate of decrease in the corresponding system unbalance. The fitness function in this case can be represented as:

$$\text{Maximize } f_2 = \frac{\text{TH}_{\text{seq}} - \text{TH}_{\min}}{\text{TH}_{\max} - \text{TH}_{\min}}, \tag{2}$$

where

$\text{TH}_{\max}$ is the maximum throughput,
$\text{TH}_{\min}$ is the minimum throughput (= 0 units),
$\text{TH}_{\text{Seq}}$ is the throughput corresponding to a particular part type sequence.

(3) Combination of minimization of system unbalance and maximization of throughput: Mukhopadhyay *et al.* (1992), Tiwari *et al.* (1997) addressed machine loading problems with an objective of minimizing system unbalance and thereby maximizing throughput. In the case of overloading of the machine, naturally the throughput will get enhanced and this will lead to an increase in system unbalance, whereas in the case of underloading of machines, throughput will decrease and system unbalance will increase. This combined objective function yields a part type sequence and operation-machine allocation such that a decrease in system unbalance is accompanied by a substantial increase in throughput. To take care of these facts, the following fitness function has been suggested:

$$\text{Maximize } f_3 = \frac{W_1 \times [f_1] + W_2 \times [f_2]}{W_1 + W_2}, \tag{3}$$

where

$f_1$ $= [(SU_{max} - SU_{seq})/(SU_{max} - SU_{min})]$,
$f_2$ $= [(TH_{seq} - TH_{min})/(TH_{max} - TH_{min})]$,
$W_1$ is the weightage assigned to objective 1 (minimization of the system unbalance),
$W_2$ is the weightage assigned to objective 2 (maximization of throughput),
$W_1 = W_2$ $= 1$ (for our case).

Mukhopadhyay *et al.* (1992) and Tiwari *et al.* (1997) have adopted heuristic solutions approaches to machine loading problems where the part sequence was predetermined (based on the SPT sequence). Their research focused on allocating the operations of the part type on different machines so as to minimize the system unbalance and thereby maximize the throughput by satisfying the system constraints, such as availability of machining time and number of tool slots. However, in this research, we generate an initial population of part sequences randomly and these sequences are evaluated with respect to the above-mentioned objective functions by taking into account similar system constraints. The various steps that are required to evaluate the part type sequence using an objective function are as follows:

*Step* 1. Input the total number of machines $m_{max}$, where '$m$' is the index for machines, $m = 1,2,\ldots,m_{max}$. For $m = 1$ to $m_{max}$, input the $Atm_m$ and $Ats_m$, where, $Atm_m$ = Available machining time on machine '$m$', $Ats_m$ = Available tools slots on machine '$m$'.

*Step* 2. Input the part type sequence having $j_{max}$ part types arranged at $k_{max}$ positions, where '$j$' is the index for part type number, $j = 1$ to $j_{max}$, and '$k$' is the index for the position of the part type in the sequence, $k = 1$ to $k_{max}$.

*Step* 3. Initialize $k = 1$, and $o = 1$.

*Step* 4. According to the input sequence, for the part type '$j$' at the $k$th position,
**If** the part type contains the essential operation,
**Then** for the essential operation '$o$', where '$o$' is the index for operation number, $o = 1$ to $o_{max}$, determine $etm_{jom}$, $ets_{jom}$, $Atm_m$, $Ats_m$, where $etm_{jom}$ is the essential time requirement of part type '$j$' for operation '$o$' on machine '$m$', $ets_{jom}$ is the essential tool slot requirement of part type '$j$' for operation '$o$' on machine '$m$',
**Else** go to step 8.

*Step* 5. If $Ats_m \geq ets_{jom}$,
**Then** go to step 6,
**Else** reject the part type '$j$' due to tool slot constraints (TSC) and increase '$k$' by 1 ($k = k + 1$), and go to step 4.

*Step* 6. **If** $Atm_m \geq etm_{jom}$,
**Then** allocate the essential operation '$o$' of the part type '$j$' on machine '$m$'. Determine $Rtm_m$, $Rts_m$, where $Rtm_m$ is the remaining time on the machine '$m$' after the allocation and $Rts_m$ is the remaining tool slot on machine '$m$' after allocation. Set $Atm_m = Rtm_m$, and $Ats_m = Rts_m$.
**Else** determine $SUA_{jom}$, where $SUA_{jom}$ is the system unbalance after allocating the operation '$o$' of the part type '$j$' on machine '$m$'.
**If** $SUA_{jom}$ is negative,

**Then** reject the part type '$j$' due to Negative System Unbalance (NSU). Increase $k$ by 1 ($k = k + 1$),
Go to step 4.
**Else** allocate the essential operation '$o$' of the part type '$j$' on machine '$m$'. Determine $\mathrm{Rtm}_m$ and $\mathrm{Rts}_m$ and set $\mathrm{Atm}_m = \mathrm{Rtm}_m$ and $\mathrm{Ats}_m = \mathrm{Rts}_m$.

*Step* 7. **If** $o < o_{\max}$,
**Then** increase '$o$'' by 1 ($o = o + 1$) and go to step 6,
**Else** go to step 8.

*Step* 8. Determine $\mathrm{otm}_{jom}$ for all the optional operations '$o$', $o = 1$ to $o_{\max}$, $\mathrm{otm}_{jom}$ is the machining time of the optional operation '$o$' of part type '$j$' on machine '$m$' and $\mathrm{ots}_{jom}$ is the tool slots required for carrying out the optional operation '$o$' of part type '$j$' on machine '$m$'.

*Step* 9. Initialize $o = 1$.

*Step* 10. **If** max $\mathrm{Atm}_m \geq \mathrm{otm}_{jom}$, where max $\mathrm{Atm}_m$ is the maximum available time associated with machine '$m$',
**Then** go to step 11,
**Else** evaluate $\mathrm{SUA}_{jom}$.
**If** $\mathrm{SUA}_{jom}$ is positive,
**Then** go to step 11,
**Else** reject the part type due to negative system unbalance, and set $k = k + 1$ and go to step 4.

*Step* 11. **If** ($\mathrm{Ats}_m$ corresponding to Max $\mathrm{Atm}_m$) $\geq \mathrm{ots}_{jom}$,
**Then** allocate the optional operation '$o$' of the part type '$j$' on machine '$m$'. Determine $\mathrm{Rtm}_m$, $\mathrm{Rts}_m$ after allocation. Set $\mathrm{Atm}_m = \mathrm{Rt}_m$ and $\mathrm{Ats}_m = \mathrm{Rts}_m$. Go to step 12.
**Else** reject the part type '$j$' due to tool slot constraints and increase $k$ by 1 ($k = k + 1$) and go to step 4.

*Step* 12. If $o < o_{\max}$,
**Then** increase '$o$' by 1 ($o = o + 1$) and go to step 10,
**Else** go to step 13.

*Step* 13. **If** $k < k_{\max}$,
**Then** go to step 2,
**Else** go to step 14.

*Step* 14. Determine System Unbalance 'SU', Throughput 'TH', Set of part types unassigned due to tool slot constraints (TSC) '$\mathrm{JR}_{\mathrm{NSU}}$', set of part types unassigned due to negative system unbalance (NSU) '$\mathrm{JR}_{\mathrm{TSC}}$'.

$$\mathrm{SU} = \sum_{m=1}^{m_{\max}} \mathrm{Atm}_m \text{ after allocation of the part type at '} j \text{'}$$

that $j$ corresponds to '$k_{\max}$'th position in the sequence.

$$\mathrm{TH} = \sum_{j=1}^{j_{\max}} \text{Batch size of the part type } j\left(j \notin \mathrm{JR}_{\mathrm{NSU}}, j \notin \mathrm{JR}_{\mathrm{TSC}}\right)$$

*Step* 15. Select the objective function $(f_1, f_2, f_3)$ and evaluate the fitness function value.

### 4.4. *Selection*

Selection models nature's survival-of-the-fittest mechanism. Fitter solutions survive while weaker ones perish. In the simple genetic algorithm, the fittest string receives a higher number of offspring and thus has a higher chance of surviving in the subsequent generation. Back (1994) proposed many selection schemes to achieve a balance between population diversity and selective pressure. Goldberg *et al.* (1989) suggested the tournament selection in messy GAs and this has been most commonly utilized to provide good solutions.

In this research, the selection scheme practised is able to determine the fittest part type sequence and is given as follows.

The expected count of each part type sequence $e_i$ is calculated as $e_i = (f_i / \sum f_i) \times \text{POP\_SIZ}$, where $f_i$ is the fitness value of the $i$th sequence and ($f_i$ is the sum of fitness value of all the sequences in the population, and POP_SIZ is the population size. Each sequence is then allocated to samples according to the integer part of ei values and the functional part of ei is treated as a probability. For example, if there are four sequences in a population having objective function values as given below:

| Sequences | Obj. fun. value ($f_i$) | $f_i / \sum f_i$ | $e_i$ | RWC |
|---|---|---|---|---|
| [3 4 6 2 1 5] | 338 | 0.144 | 0.576 | 1 |
| [3 4 2 1 5 6] | 1152 | 0.429 | 1.960 | 2 |
| [3 4 1 5 2 6] | 128 | 0.054 | 0.216 | 0 |
| [6 2 5 1 4 3] | 722 | 0.308 | 1.230 | 1 |
| | 2340 | | | |

Holland (1975) used a roulette wheel selection scheme to implement the proportionate selection, which is similar to the above-discussed method. Similarly, Goldberg (1989) also mentioned a similar selection scheme and used the term 'actual count of roulette wheel' (RWC), which is the next higher integer value of the expected count. Therefore, the next generation will consist of a copy of the first sequence [3 6 4 2 1 5], two copies of the second sequence [3 4 2 1 5 6], and a copy of fourth sequence [6 2 5 1 4 3]. The third sequence [3 4 1 5 2 6] dies off.

### 4.5. *Crossover and mutation*

*Crossover*: Crossover is a process by which two parent strings recombine to produce two new offspring strings. The standard crossover operators that are most commonly used in solving sequencing and scheduling problems include partially mapped crossover (PMX) (Goldberg *et al.* 1985), enhanced edge recombination (EER) (Starkwather *et al.* 1991), order crossover (OX) (Davis 1985b), uniform order-based crossover (UOX) (Davis 1991), cycle crossover (CX) (Oliver *et al.* 1987), etc. These crossover operators operate on the representations of ordering or the permutation of elements. It is to be noted that crossover is not always effected. After choosing a pair of strings, the algorithm evokes crossover only if a randomly generated number in the range of 0 to 1 is greater than the crossover rate (it is also known as crossover probability), otherwise the string remains unaltered. In this

research, partially mapped crossover operators has been applied. For example, consider the parent string $P_1 = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8]$ and $P_2 = [4\ 8\ 7\ 3\ 2\ 1\ 5\ 6]$ chosen for crossover. In applying the PMX operator to $P_1$ and $P_2$, first randomly choose a common interval from $P_1$ and $P_2$. In this example, an interval from positions four to seven is randomly chosen. Then determine the mapping of the elements in the two selected intervals. In this case, the mappings between the selected intervals are 3 to 4, 2 to 5, and 1 to 6. Then, swap the two intervals in $P_1$ and $P_2$. Now, to ensure that it is a feasible sequence (i.e. the repetition of part type is avoided), copy from the left the chromosomes of $P_1$ to yield offspring $O_1$ and $P_2$ to yield offspring $O_2$ by excluding the elements in the intervals.

$$P_1 = [1\ 2\ 3|4\ 5\ 6|7\ 8]$$

$$P_2 = [4\ 8\ 7|3\ 2\ 1|5\ 6]$$

$$\qquad\qquad\ \ |\ \ |\ \ |$$

$$O_1 = [4\,5\,6|3\ 2\ 1|7\ 8]$$

$$O_2 = [3\ 8\ 7|4\ 5\ 6|2\ 1]$$

However, in this research, we have also tested other crossover operators such as order crossover (OX), cycle crossover (CX), uniform order-based crossover (UOX), enhanced edge recombination (EER).

*Mutation*: After crossover, normally strings are subjected to mutation. Mutation randomly alters the composition of a string to produce a new offspring instead of recombining two strings. In traditional genetic algorithms, mutation of a bit involves flipping it: changing a '0' to '1' or vice versa. In the case of mutation, a parameter known as a mutation rate gives the probability with which a bit will be flipped. The bits of the string are independently muted, i.e. the mutation of the bit does not influence the probability of mutation of another bit. Herdy (1991) proposed four different mutation operators, namely insertion (INS), reciprocal exchange (RE), inversion (INV), and displacement (DIS). In genetic algorithms, mutation empowers it to explore the search space. The mutation operator used in this research is similar to reciprocal exchange, which randomly selects two positions in the string and swaps the part types in these two positions to generate a new string. For instance, when applying the mutation operation to the string: $P_1 = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8]$, it randomly picks up two positions of the string (position 2 and position 6) and swaps the part types in these positions (part types 2 and 6) to generate a new string $O_1 = [1\ 6\ 3\ 4\ 5\ 2\ 7\ 8]$.

However, in this research, we have also tested the effect of other mutation operators such as inversion (INV), insertion (INS), and displacement (DIS).

### 4.6. *Reproduction*

After the crossover and mutation operations have been performed, we are left with the generation consisting of initial population, crossover offsprings and mutation offsprings. Out of this extended population, POP_SIZ number of chromosomes (equal to the size of initial population) have to be entered in the next generation. Several researchers have used roulette wheel selection as the basic principle of reproduction in GAs, where the chromosomes are selected with probability biased towards a string with better evaluation. Liepins and Hilliard (1989) suggested several replacement strategies in order to replace the poor performing offspring in the new

generations. They also mentioned that the elitist strategy appends the best performing chromosome of the previous generation to the current population and thereby gets an assurance that the sequences with best performance always survive in the next generation. In this paper, the elitist way was embedded within the roulette wheel selection in order to enforce the preservation of the best chromosome in the next generation and overcome the stochastic errors of sampling. For example, out of the extended population, the string having the highest fitness function is carried to the next generation and roulette wheel selection is to spin (POP_SIZ-1) times to get the other strings of the next generation.

### 4.7. *Selection of GA parameters*

The most difficult and time-consuming issue in the successful operation of GAs is determining good parameter settings. These parameters are: crossover probability (PC), mutation probability (PM), population size (POP_SIZ), number of generation (MAX_GEN), etc. Grefenstette (1986) discussed the importance of choosing the appropriate values for these parameters of the optimization algorithm, which needs to be tuned for efficiency. However, Michalewicz (1992) mentioned that the determination of proper values of these genetic parameters still remains an art rather than science. In this paper, we have conducted extensive experiments with different combinations of parameter values. The ranges of exact values of these parameters have been specified in the illustrative examples.

## 5. An illustrative example

In this research, a genetic-algorithm-based heuristic approach has been developed and applied to solve the example problems considered by Mukhopadhyay *et al.* (1992), and Tiwari *et al.* (1997) for a random type FMS. Table 1 shows the detail description of problem number 1. Mukhopadhyay *et al.* (1992) generated nine more problems, which have also been solved using the proposed heuristic. The procedure of the proposed GA based heuristic for solving the machine loading problem is given as follows:

*Step* 1. Set initial and global parameters. Set population size POP_SIZ, crossover probability PC, mutation probability PM, and maximum generation MAX_GEN. Set initial generation GEN = 0 and initial fitness function value = 0.

*Step* 2. Generate the initial population. Generate initial population by randomly arranging $j_{\max}$ genes in a sequence, where $j_{\max}$ is the total number of part types to be loaded on machines in a shift.

*Step* 3. Select an objective function $(f_1, f_2, f_3)$ given by equations (1), (2) and (3) respectively, depending upon the type of the problem) for the evaluation of the fitness of the strings.

*Step* 4. Set GEN = GEN + 1. Evaluate the initial population as per the objective function selected using the steps given in section 4.3.

*Step* 5. Select the strings in the initial population for crossover and mutation (as per selection scheme suggested in section 4.4).

*Step* 6. Perform crossover and mutation operators on the initial population to generate new offsprings (as suggested in section 4.5).

*Step* 7. Evaluate the extended population of strings (using the reproduction strategy suggested in section 4.6) and select the strings for the next generation.

| Part type | Operation no. | Batch size | Unit processing time | Machine number | Tool slot needed |
|---|---|---|---|---|---|
| 1 | 1 | 8 | 18 | 3 | 1 |
| 2 | 1 | 9 | 25 | 1 | 1 |
| | | | 25 | 4 | 1 |
| | 2 | | 24 | 4 | 1 |
| | 3 | | 22 | 2 | 1 |
| 3 | 1 | 13 | 26 | 4 | 2 |
| | | | 26 | 1 | 2 |
| | 2 | | 11 | 3 | 3 |
| 4 | 1 | 6 | 14 | 3 | 1 |
| | 2 | | 19 | 4 | 1 |
| 5 | 1 | 9 | 22 | 2 | 2 |
| | | | 22 | 3 | 2 |
| | 2 | | 25 | 2 | 1 |
| 6 | 1 | 10 | 16 | 4 | 1 |
| | 2 | | 7 | 4 | 1 |
| | | | 7 | 2 | 1 |
| | | | 7 | 3 | 1 |
| | 3 | | 21 | 2 | 1 |
| | | | 21 | 1 | 1 |
| 7 | 1 | 12 | 19 | 3 | 1 |
| | | | 19 | 2 | 1 |
| | | | 19 | 4 | 1 |
| | 2 | 13 | 2 | 1 | 1 |
| | | | 13 | 3 | 1 |
| | | | 13 | 1 | 1 |
| | 3 | | 23 | 4 | 3 |
| 8 | 1 | 13 | 25 | 1 | 1 |
| | | | 25 | 2 | 1 |
| | | | 25 | 3 | 1 |
| | 2 | | 7 | 2 | 1 |
| | | | 1 | 1 | 1 |
| | 3 | | 24 | 1 | 3 |

Table 1. Description of Problem No. 1 (adopted from Shanker and Srinivasulu 1989).



*Step* 8. Termination scheme. If GEN < MAX_GEN, return to step 4. If GEN = MAX_GEN, then output the maximum value of the objective function and the corresponding part sequence and terminate.

The application of the above GA-based heuristic for solving the machine loading problem given in table 1 has been illustrated with few of the steps, as follows.

*Step* 1. Set the initial and global parameters: POP_SIZ = 5, PC = 0.5, PM = 0.2, MAX_GEN = 50, GEN = 0, $f_1 = 0$.

*Step* 2. Total number of part types = $J_{max} = 8$, and part type sequences randomly generated are as follows:

[8 3 4 6 7 5 1 2], [5 4 3 7 1 6 8 2], [5 4 3 2 1 6 8 7], [4 8 5 2 1 3 6 7], and [7 5 8 4 2 1 3 6].

*Step* 3. Objective function: $f_1$. Minimization of system unbalance.

*Step* 4. Set GEN $= 0 + 1 = 1$ and evaluate the part type sequences generated in step 2. For the string [8 3 4 6 7 5 1 2], the value of $f_1$ is determined (using steps given in section 4.3) as follows.

*Step* 1. $m = 4$, $\mathrm{Atm}_1 = \mathrm{Atm}_2 = \mathrm{Atm}_3 = \mathrm{Atm}_4 = 480$ min. and $\mathrm{Ats}_1 = \mathrm{Ats}_2 = \mathrm{Ats}_3 = \mathrm{Ats}_4 = 5$.

*Step* 2. Part type sequence = 8, 3, 4, 6, 7, 5, 1, 2 and $k = 1{:}j = 8$, $k = 2{:}j = 3$, $k = 3{:}j = 4, \ldots, k = 8{:}j = 2$, and $k_{\max} = j_{\max} = 8$.

*Step* 3. Set $k = 1$, $o = 1$.

*Step* 4. For $k = 1$, $j = 8$, $o = 1$, $\mathrm{etm}_{811} = 312$, $\mathrm{ets}_{811} = 3$, $\mathrm{Atm}_1 = \mathrm{Atm}_2 = \mathrm{Atm}_3 = \mathrm{Atm}_4 = 480$ and $\mathrm{Ats}_1 = \mathrm{Ats}_2 = \mathrm{Ats}_3 = \mathrm{Ats}_4 = 5$.

*Step* 5. $\mathrm{Ats}_1 > \mathrm{ets}_{811}$.

*Step* 6. $\mathrm{Atm}_1 > \mathrm{etm}_{811}$, hence allocate essential operation of part type on machine 1.
$\mathrm{Rtm}_1 = 168$, $\mathrm{Rts}_1 = 2$, $\mathrm{Atm}_1 = 168$, $\mathrm{Atm}_2 = \mathrm{Atm}_3 = \mathrm{Atm}_4 = 480$ and $\mathrm{Ats}_1 = 2$, $\mathrm{Ats}_2 = \mathrm{Ats}_3 = \mathrm{Ats}_4 = 5$.

*Step* 7. $o = o_{\max}$.

*Step* 8. For optional operation $o = 1$, $\mathrm{otm}_{811} = \mathrm{otm}_{812} = \mathrm{otm}_{813} = 325$ and $\mathrm{ots}_{811} = \mathrm{ots}_{812} = \mathrm{ots}_{813} = 1$.
For optional operation $o = 2$, $\mathrm{otm}_{821} = \mathrm{otm}_{822} = 91$
And $\mathrm{ots}_{821} = \mathrm{ots}_{822} = 1$.

*Step* 9. Set $o = 1$.

*Step* 10. $(\mathrm{Max}\ \mathrm{Atm}_3 = \mathrm{Max}\ \mathrm{Atm}_2) > \mathrm{otm}_{812}$.

*Step* 11. $\mathrm{Ats}_2 > \mathrm{ots}_{812}$, hence allocate to machine 2 and $\mathrm{Rtm}_2 = 155$, $\mathrm{Rts}_2 = 4$, $\mathrm{Atm}_1 = 168$, $\mathrm{Atm}_3 = \mathrm{Atm}_4 = 480$, $\mathrm{Atm}_2 = 155$, and $\mathrm{Ats}_1 = 2$, $\mathrm{Ats}_3 = \mathrm{Ats}_4 = 5$, $\mathrm{Ats}_2 = 4$.

*Step* 12. $o < o_{\max}$, set $o = 2$.

*Step* 13. $\mathrm{Max}\ \mathrm{Atm}_1 > \mathrm{otm}_{821}$.

*Step* 11. $\mathrm{Ats}_1 > \mathrm{ots}_{821}$, hence allocate to machine 1 and $\mathrm{Rtm}_1 = 77$, $\mathrm{Rts}_1 = 1$, $\mathrm{Atm}_1 = 77$, $\mathrm{Atm}_3 = \mathrm{Atm}_4 = 480$, $\mathrm{Atm}_2 = 155$, and $\mathrm{Ats}_1 = 1$, $\mathrm{Ats}_3 = \mathrm{Ats}_4 = 5$, $\mathrm{Ats}_2 = 4$.

*Step* 12. $o = o_{\max}$, and

*Step* 13. $k < k_{\max}$, and $k = k + 1$.

*Step* 4. For $k = 2$, $j = 3$ and allocate essential operation of part type 3, ... on machine 3.

*Step* 5. ...

*Step* 6. ...

*Step* 7. ...

*Step* 8. ....

*Step* 9. ...

*Step* 10. ...

*Step* 11. Allocate the optional operations of part type 3 on machine 4.

*Step* 12. ...

*Step* ...

The summary of allocation of the remaining part types is given below:

Part type 4: Operation 1, Machine 3, Tool slot 1,
Part type 4: Operation 2, Machine 4, Tool slot 1,
Part type 6: Operation 1, Machine 4, Tool slot 1,
Part type 6: Operation 2, Machine 3, Tool slot 1,
Part type 6: Operation 3, Machine 2, Tool slot 1,
Part type 7: Operation 3, Machine 4, Tool slot 3, and NSU,
Part type 5: Operation 2, Machine 2, Tool slot 1, and NSU,
Part type 1: Operation 1, Machine 3, Tool slot 1, and NSU,
Part type 2: Operation 2, Machine 4, Tool slot 1, and NSU.
For the part type sequence [8 3 4 6 7 5 1 2], SU = 73 and TH = 42,
$JR_{NSU} = (7,5,1,2)$, $JR_{TSC} = (\phi)$, the value of objective function = 0.961.

*Step* 5.

| Sequence | SU | TH | $f_i$ | $[f_i/\sum f_i]$ | $E_I$ | RWC |
|---|---|---|---|---|---|---|
| [8 3 4 6 7 5 1 2] | 73 | 42 | 0.962 | 0.208 | 0.834 | 1 |
| [5 4 3 7 6 1 8 2] | 55 | 45 | 0.971 | 0.210 | 0.842 | 1 |
| [5 4 3 2 1 6 8 7] | 76 | 42 | 0.960 | 0.208 | 0.832 | 1 |
| [4 8 5 2 1 3 6 7] | 427 | 36 | 0.778 | 0.146 | 0.674 | 1 |
| [7 5 8 4 2 1 3 6] | 109 | 34 | 0.943 | 0.204 | 0.817 | 1 |

$\sum f_i = 4.614$

All the part sequences are selected for crossover and mutation.

*Step* 6. Performed PMX crossover on randomly selected strings
$P_1 = [4\ 8\ 5|2\ 1\ 3|6\ 7]$ and $P_2 = [7\ 5\ 8|4\ 2\ 1|3\ 6]$
Yielding offsprings $O_1 = [8\ 5\ 3\ 4\ 2\ 1\ 6\ 7]$ and $O_2 = [7\ 5\ 8\ 2\ 1\ 3\ 4\ 6]$.
Performed RE mutation on $P_3 = [5\ 4\ 3\ 2\ 1\ 6\ 8\ 7]$
yielding $O_3 = [6\ 4\ 3\ 2\ 1\ 5\ 8\ 7]$.
Similarly, the crossover and mutation was performed on other strings to extend the population.

*Step* 7. The extended population consists of initial population selected for reproduction + crossover offsprings + mutation offsprings. Evaluate the extended population using the objective function $f_1$ (steps are given in section 4.3). Here, the extended population consists of nine strings, out of which the best one is preserved in the next generation (as per Elitist Strategy). The remaining four strings are selected from the remaining eight using roulette wheel count.

*Step* 8. GEN < MAX_GEN$(1 < 50)$.

*Step* 4. Set GEN = GEN + 1 = 2. Evaluate the five strings of the GEN 1 and evaluate for reproduction.

*Step* 5. ...

*Step* 6. ...

*Step* 7. ...

*Step* 8. GEN = MAX_GEN and terminate and output the result.
Part sequence = [5 7 4 2 1 3 6 8]
SU = 0 min, TH = 36 units, $f_1 = 1.0$, $JR_{NSU} = (1,3,6,8)$ and $JR_{TSC} = (\phi)$.

## 6. Results and discussions

The proposed GA-based heuristic employs three criteria, namely system unbalance alone, throughput alone and a combination of system unbalance and throughput to justify the result obtained. It can easily be correlated that the minimization of system unbalance leads to the maximization of capacity utilization of the system. At this point, we have to take into account the overloading and underloading of the machines and how such measures have been incorporated in the proposed solution methodology. Shanker and Srinivasulu (1989) have taken only unutilized machine time in the system unbalance whereas Mukhopadhyay *et al.* (1992) and Tiwari *et al.* (1997) have considered the unutilized and overutilized machine time in their system unbalance. It is worth mentioning here that the heuristic proposed by Tiwari *et al.* (1997) has considered the allocation of next part type from the set of unassigned part type by removing the last assigned part type when the system unbalance starts increasing. This happens when the actual system unbalance is found negative.

Mukhopadhyay *et al.* (1992) have adopted the max-max rule for the allocation of an unsigned part type in anticipation that the system unbalance improves further. The rule suggests that the last part type of the allocated set of part types be reallocated by reallocating the maximum remaining time of the machine to the maximum operation time, or processing time, of each operation of the selected last part type. The purpose of reallocation of part types in the heuristic is to avoid the situation in which improvements in the throughput are accompanied by an increase in system unbalance. However, the proposed GA-based heuristic takes care of the above situation by having a choice of different fitness functions ($f_1$, $f_2$ and $f_3$). Most of the previous researchers have concluded that the performance of the 'shortest processing time (SPT)' sequencing rule for loading the part type on machines works, on average, better. However, in this research, we have analysed the machine loading problem using the heuristic proposed by Tiwari *et al.* (1997) with several predetermined sequencing rules, such as LPT (Longest Processing Time), LIFO (Last In First Out), FIFO (First In First Out), with the objective function of minimizing system unbalance and thereby maximizing throughput. And we have compared the results using the GA-based heuristic proposed in this paper. The results are shown in table 2. From this table, it is clear that proposed GA-based heuristic outperforms the results obtained with fixed predetermined sequencing rules. The proposed GA-based heuristic takes care of the above facts by the formulation of different fitness functions.

In this research, the part type sequence has been determined using the random search technique and, after performing several trials, the solutions obtained are better than the SPT-based sequence of part types. For the problem given in table 1, the proposed GA-based heuristic with the objective of minimizing system unbalance and maximizing throughput yields a system unbalance of 14 minutes and a throughput of 48 units, whereas Tiwari *et al.* (1997) arrived at a system unbalance of 76 and a throughput of 42. For the same problem, Mukhopadhyay *et al.* (1992) reported a system unbalance of 122 and a throughput of 42, whereas Shanker and Srinivasulu (1989) gave a system unbalance of 253 and a throughput of 39. In all the above research, same technological constraints, i.e. the availability of machining

| Problem no. | SPT sequence | | LPT sequence | | LIFO sequence | | FIFO sequence | | Sequencing using GA-based heuristic | |
|---|---|---|---|---|---|---|---|---|---|---|
| | SU | TH | SU | TH | SU | TH | SU | TH | SU | TH |
| 1 | 76 | 42 | 51 | 38 | 92 | 35 | 35 | 45 | 14 | 48 |
| 2 | 236 | 63 | 18 | 46 | 154 | 63 | 500 | 57 | 18 | 46 |
| 3 | 152 | 69 | 132 | 48 | 72 | 69 | 152 | 69 | 72 | 69 |
| 4 | 819 | 51 | 819 | 51 | 818 | 51 | 819 | 51 | 819 | 51 |
| 5 | 264 | 61 | 187 | 53 | 211 | 53 | 207 | 52 | 187 | 53 |
| 6 | 314 | 63 | 139 | 49 | 314 | 63 | 500 | 57 | 28 | 61 |
| 7 | 996 | 48 | 165 | 54 | 996 | 48 | 189 | 54 | 165 | 54 |
| 8 | 158 | 43 | 133 | 29 | 13 | 44 | 158 | 43 | 63 | 48 |
| 9 | 309 | 88 | 309 | 88 | 309 | 88 | 309 | 88 | 309 | 88 |
| 10 | 166 | 55 | 184 | 49 | 82 | 54 | 166 | 55 | 122 | 56 |
| Average | 349 | 58.3 | 213.7 | 50.5 | 306.1 | 56.8 | 303.5 | 57.1 | 179.7 | 57.4 |

SU. system unbalance; TH: througput.

Table 2. Comparison of GA based heuristic with that of heuristic proposed by Tiwari *et al.* (1997) (using different sequencing rules).



time and tool slots on machines are taken into account. Similarly, for all the other nine problems generated by Mukhopadhyay *et al.* (1992), a comparative study has been carried out and results are reported in table 3. It is evident from table 3 that the proposed GA-based heuristic is the most efficient one. Detailed enumeration of the parameters for ten problems — such as the choice of objective function, GA control parameter settings, part type sequence, fitness function value, corresponding system unbalance and throughput, and the set of part types unassigned (along with causes of rejection) etc — are given in table 4. Apart from showing substantial improvement in the solution quality, the proposed heuristic takes care of the weakness of other fixed predetermined sequencing-rule based heuristics.

So far, we have discussed the application of a genetic-algorithm based heuristic in solving the machine loading problem and have enumerated the objective function and other related parameters for ten sample problems. However, the coming section discusses the difficulty and time-consuming issues, i.e. finding good parameter settings for a genetic algorithm that has been successfully implemented with regard to the machine-loading problem. Davis (1991) suggested a number of approaches to derive robust parameter setting for Gas, including carrying out brute force searches using an adaptive operator fitness technique. De Jong (1975) was the first to study the importance of choosing the appropriate values for various genetic parameters; namely, crossover rate, mutation rate, population size, number of generation, replacement strategies etc. Schaffer *et al.* (1989) found that optimal values for these parameter settings change from problem to problem. To determine systematically appropriate parameter values, a decision-support-system based approach was proposed by Pakath and Zaveri (1993) for solving a given problem. The experimental design approach to evaluate the setting of several GA parameters was discussed by Gupta *et al.* (1993a, 1993b). There are several possible elaborations of a genetic algorithm involving variations in parameters such as size of population, crossover

| Problem no. | Total no. of part types | Shanker and Srinivasulu (1989) | | Mukhopadhyay *et al.* (1992) | | Tiwari *et al.* (1997) | | Proposed GA-based heuristic | |
|---|---|---|---|---|---|---|---|---|---|
| | | SU | TH | SU | TH | SU | TH | SU | TH |
| 1 | 8 | 253 | 39 | 122 | 42 | 76 | 42 | 14 | 48 |
| 2 | 6 | 388 | 51 | 202 | 63 | 234 | 63 | 18 | 46 |
| 3 | 5 | 288 | 63 | 286 | 79 | 152 | 69 | 72 | 69 |
| 4 | 5 | 819 | 51 | 819 | 51 | 819 | 51 | 819 | 51 |
| 5 | 6 | 467 | 62 | 364 | 76 | 264 | 61 | 187 | 53 |
| 6 | 6 | 548 | 51 | 365 | 62 | 314 | 63 | 28 | 61 |
| 7 | 6 | 189 | 54 | 147 | 66 | 996 | 48 | 165 | 54 |
| 8 | 7 | 459 | 36 | 459 | 36 | 158 | 43 | 63 | 48 |
| 9 | 7 | 462 | 79 | 315 | 88 | 309 | 88 | 309 | 88 |
| 10 | 6 | 518 | 44 | 320 | 56 | 166 | 55 | 122 | 56 |

Table 3. Comparison of the result obtained using proposed GA based heuristic with other heuristic solutions.



rate, mutation rate, selection strategies etc. The choice of optimal control parameter has been discussed in detail (either by using analytical methods or by empirical investigations). One cannot choose control parameters until the interactions among the genetic operations are considered in detail. It is because these parameters cannot be determined independently, that the choice of control parameter is a complex nonlinear optimization issue. Moreover, it is increasingly clear that the optimal control parameters critically depend on the nature of the objective function. Therefore, the trade-offs that arise are to be pointed out in order to select the optimal value of control parameters. These points are as follows.

- Whenever the crossover probability increases, it leads to an increase in the recombination of the building blocks. However, it is accompanied by disruption of good strings.
- In the case where the mutation probability increases, it leads to a transformation of the genetic search into random search, but it is also accompanied by a reproduction of large genetic search material.
- Increasing the population size leads to an increase in its diversity and reduces the probability that the GA will prematurely converge to local optimal results; however, it also increases the time requirement to converge to the optimal region in the search space.

Therefore, it should be noted that tuning the parameter values of GA is a complex process and can affect the efficiency of the algorithm. In most applications of GA, these parameter values are tuned based on some trial examples. The ten problems tested consist of 5 to 8 part types. The maximum number of genes in the chromosomes representing the part type sequence ranges from 5 to 8. Several trials were conducted from population size 5 to 15 in step 2 and from 15 to 30 in steps of 5. Similarly, the crossover probability was varied from 0 to 1 in steps of 0.1, and the mutation probability from 0 to 1 in steps of 0.1.

The problems were also attempted by setting the cross probability to 0 and by varying the mutation probability and vice versa, in order to see the diversity of the objective function values. The maximum generation for these problems was varied

| Pr. No. | Ob. Fu. | Parameter settings of GA/PS | Part type sequence | Part type unassigned | | Value of objective function | SU | TH |
|---|---|---|---|---|---|---|---|---|
| | | | | NSU | TSC | | | |
| 1 | 1 | 5, 0.5, 0.2, 50 | 5, 7, 4, 2, 6, 3, 1, 8 | 6, 3, 1, 8 | — | 1.000 | 0 | 36 |
| | 2 | 5, 0.5, 0.2, 50 | 4, 7, 3, 2, 1, 6, 8, 5 | 2, 6 | 8 | 0.640 | 14 | 48 |
| | 3 | 5, 0.5, 0.2, 50 | 4, 7, 3, 1, 2, 6, 8, 5 | 2, 6 | 8 | 0.796 | 14 | 48 |
| 2 | 1 | 5, 0.7, 0.2, 50 | 6, 2, 5, 3, 4, 1 | 3, 1 | — | 0.919 | 18 | 46 |
| | 2 | 5, 0.7, 0.2, 50 | 4, 5, 6, 1, 2, 3 or 3, 5, 6, 1, 2, 4 | 2<br>— | —<br>2 | 0.630 | 154 | 63 |
| | 3 | 5, 0.5, 0.2, 50 | 3, 1, 5, 6, 4, 2 or 5, 3, 1, 4, 2, 6 | 2<br>— | —<br>2 | 0.891 | 154 | 63 |
| 3 | 1 | 5, 0.5, 0.2, 50 | 5, 4, 2, 3, 1 | 1 | — | 0.963 | 72 | 69 |
| | 2 | 5, 0.5, 0.2, 50 | 2, 3, 1, 5, 4 | 4 | — | 0.924 | 128 | 73 |
| | 3 | 5, 0.5, 0.2, 50 | 5, 1, 2, 3, 4 or 3, 5, 1, 4, 2 | 4<br>— | —<br>4 | 0.928 | 128 | 73 |
| 4 | — | —— | Any sequence | — | — | — | 819 | 51 |
| 5 | 1 | 5, 0.8, 0.2, 50 | 3, 5, 1, 4, 2, 6 | 2, 6 | — | 0.903 | 187 | 53 |
| | 2 | 5, 0.8, 0.2, 50 | 6, 2, 5, 1, 3, 4 | 4 | — | 0.815 | 479 | 62 |
| | 3 | 5, 0.8, 0.2, 50 | 5, 6, 2, 1, 4, 3 | 3 | — | 0.829 | 276 | 61 |
| 6 | 1 | 5, 0.5, 0.2, 50 | 6, 1, 4, 5, 2, 3 | 3 | — | 0.985 | 28 | 61 |
| | 2 | 5, 0.5, 0.2, 50 | 4, 3, 1, 5, 6, 2 | 2 | — | 0.863 | 314 | 63 |
| | 3 | 5, 0.5, 0.2, 50 | 1, 4, 6, 2, 5, 3 | 3 | — | 0.910 | 28 | 61 |
| 7 | 1 | 5, 0.5, 0.2, 50 | 2, 5, 4, 1, 6, 3 | 6 | 3 | 0.914 | 165 | 54 |
| | 2 | 5, 0.5, 0.2, 50 | 5, 1, 4, 3, 2, 6 and 3, 1, 2, 5, 6, 4 | 2<br>4 | —<br>— | 0.807 | 486<br>231 | 63<br>63 |
| | 3 | 5, 0.5, 0.2, 50 | 1, 5, 2, 3, 6, 4 | 4 | — | 0.843 | 231 | 63 |
| 8 | 1 | 5, 0.5, 0.2, 50 | 7, 6, 5, 4, 1, 2, 3 | 1, 2, 3 | — | 0.993 | 13 | 44 |
| | 2 | 5, 0.5, 0.2, 50 | 2, 7, 5, 4, 6, 3, 1 and 2, 7, 5, 4, 6, 1, 3 | 6, 1<br>6, 3 | —<br>— | 0.800 | 63<br>288 | 48<br>48 |
| | 3 | 5, 0.5, 0.2, 50 | 3, 2, 5, 7, 1, 6, 4 | 1, 6 | | 0.883 | 63 | 48 |
| 9 | | — | Any Sequence | — | — | — | 309 | 88 |
| 10 | 1 | 5, 0.8, 0.2, 50 | 2, 5, 4, 6, 3, 1 | 1 | — | 0.957 | 82 | 54 |
| | 2 | 5, 0.5, 0.2, 50 | 3, 5, 1, 2, 6, 4 | — | 4 | 0.835 | 122 | 56 |
| | 3 | 5, 0.5, 0.2, 50 | 1, 6, 5, 3, 2, 4 | — | 4 | 0.886 | 122 | 56 |

Pr. No.: problem number; Ob. Fu.: objective function; PS: (POP_SIZ, PC, PM, MAX_GEN); NSC: negative system unbalance; TSC: tool slot constraint.

Table 4. Details of the parameters and results obtained using proposed GA based Heuristic.

from 35 to 60 in steps of 5. It was observed that for the test problems selected, a MAX_GEN of 50 ensures the global optimal/near-optimal part sequences to be trapped.

The problems were tested using various crossover operators alone (i.e. by setting the mutation probability to 0). The crossover probability was varied and it was found that higher crossover probability yields better results. in most cases, however, premature convergence was observed. Among different crossover operators, PMX consistently showed better performance. This is evident from figure 1.

Several mutation operators were tested on their own on the given set of problems. Mutation probability was varied and it was observed that, unlike crossover, mutation did not result in premature convergence. However, the search space (POP_SIZ and MAX_GEN) is quite high. The RE operator performed better in most cases. The performance of various operators can be seen from figure 2.

Combinations of crossover and mutation operators outperform the separate use of each one of them. Optimal/near-optimal part type sequences were obtained in a very small search space (lower POP_SIZ and lesser MAX_GEN). Therefore, the joint use of crossover and mutation is desirable and recommended. In our problem, the combination of PMX and RE provided the best outcome, as these operators perform better when used alone. The performance of the combinations of various crossover and mutation operators is shown in figure 3. The results of the ten test problems are obtained using the combination of PMX-RE. The exhaustive search of parameter settings for the problem number 1 given in table 1, with the choice of objective function 3 yields the following results.

Objective function: $f_3$ Minimization of system unbalance and maximization of throughput

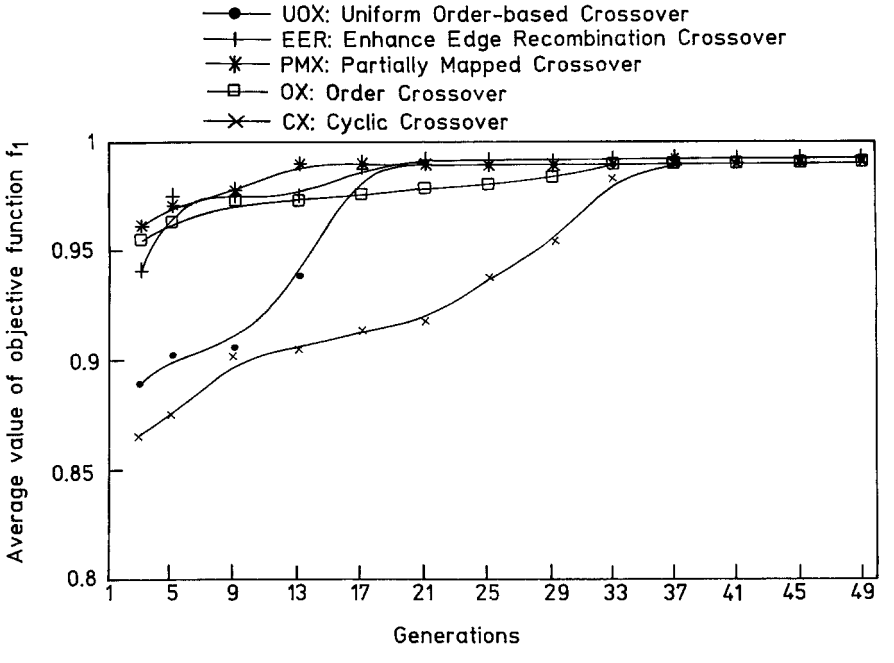


Figure 1. Performance of Crossover Operators (Problem No. 1).

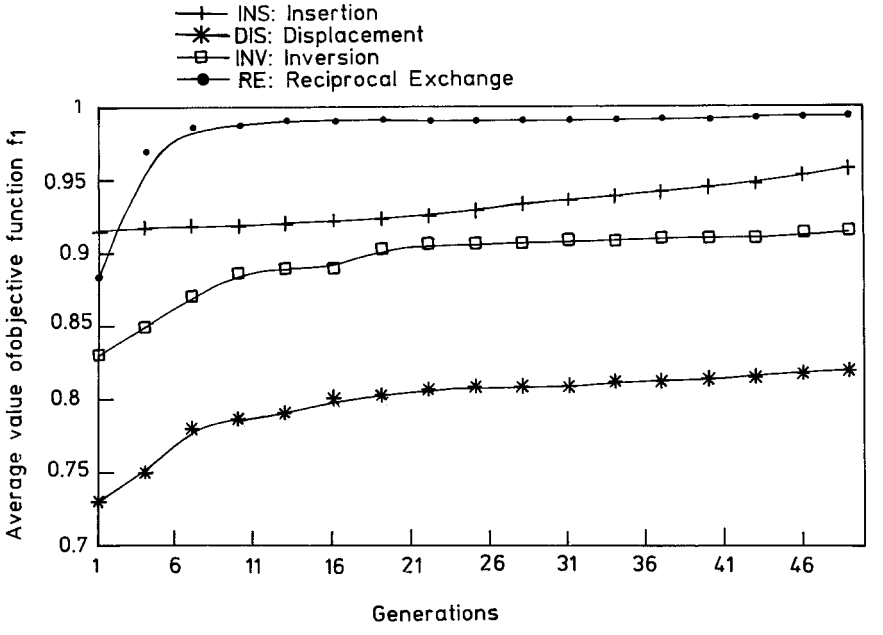


Figure 2. Performance of Mutation Operators (Problem No. 1).

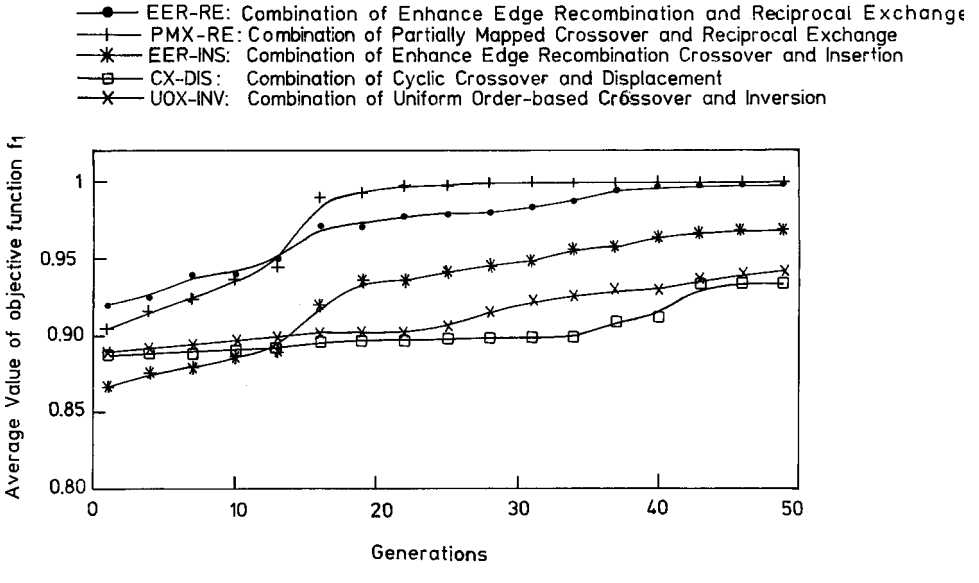


Figure 3. Performance of Combination of Operators (Problem No. 1).

Parameter settings: (POP_SIZ, PC, PM, MAX_GEN) = $(5, 0.5, 0.2, 50)$
Optimal part type sequence generated: 4, 7, 3, 8, 1, 2, 6, 5 or 4, 7, 3, 1, 2, 6, 8, 5 or 4, 7, 3, 1, 8, 6, 2, 5.
Part types unassigned: $JR_{NSU} = (2, 6)$ and $JR_{TSC} = (8)$.
Value of objective function: 0.796

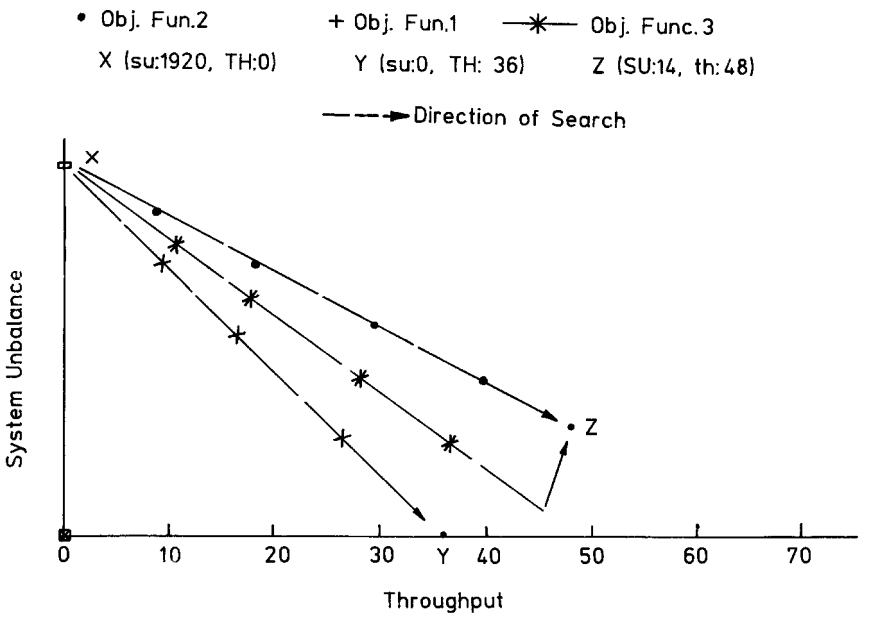


Figure 4. System Unbalance vs. Throughput ({Problem No. 1).

System unbalance = 14 minutes
Throughput = 48 units.

In order to study the effects of the combination of two objective functions, namely 'System Unbalance' and 'Throughput' which are interrelated as shown in figure 4, linear objective functions with weights assigned to the individual functions have been formulated in our problem. The $XY$ part of the curve given in the above figure illustrates the fact that when the objective is to minimize system unbalance, the GA operates to attain the point $Y$, thereby ensuring the convergence of the problem, resulting in the optimal solution. If the choice of objective function is to maximize throughput, then the GA operates to attain the point $Z$, whereas the combination of these two objectives can be treated as a multi-objective optimization problem that is governed by the assignment of variable weights. In our case, while dealing with the multiobjective function, equal weights have been taken. Therefore, the direction of search is not constant in this multiobjective formulation by GA.

The proposed GA-based heuristic has been coded in $C^{++}$ language and the program was run on an IBM PC with Pentium CPU at 133 MHz. Detailed results and analysis of the ten sample problems are available from the authors.

## 7. Conclusions

In this research, an efficient and powerful GA-based heuristic has been developed to solve the machine-loading problem in a random-type FMS. Most of the earlier researchers have addressed the loading problems using fixed sequencing rules with the objective of minimizing system unbalance and maximizing throughput by satisfying the technological constraints of the system. It was also pointed out that adopting the fixed sequencing rules for loading the part type is a weakness of the earlier

researchers' solution methodologies, as it does not guarantee a better solution to the problems. This has been overcome by proposing the GA-based heuristic, which inherently determines the part sequences that result in optimal/near-optimal solution to the problems. A comparative study has been carried out for same problem with similar objective functions and constraints and its is found that the proposed GA-based heuristic outperforms the existing heuristic that is based on solution methodologies. It is worth mentioning that the application of the GA-based heuristic developed here is limited to certain cases where there is a sufficient number of pallets, fixtures, and adequate material handling capacity.

This research and other current investigations demonstrate that the genetic-algorithm based solution methods are broad and approximate search procedures with applications in diverse problem areas. When applied to solving the machine loading problem of a random-type FMS, special emphasis is placed upon the following aspects of the GA.

(1) The representation scheme of the GA must be suitable to address the part type sequencing issues of machine loading problem of FMSs.
(2) A proper fitness function must be formulated to include the multiple objective cases.
(3) A good combination of genetic operators should be used.
(4) The most important aspect of GA application is the selection of appropriate control parameters for genetic operators, which will generate a better solution with a reasonable search space and reduced computational time.

In this research, to achieve optimal solutions, we have used several runs and trial procedures in order to decide on the values of control parameters, as follows:

$$\text{POP\_SIZ} = 5,$$
$$\text{PC} = 0.5 \text{ to } 0.8,$$
$$\text{PM} = 0.2 \text{ to } 0.4 \text{ and}$$
$$\text{MAX\_GEN} = 50.$$

This research possesses enough scope for extension to encompass several other allocations of resources, such as pallets, fixtures, AGVs etc. This research can be further extended by adding a few more objective functions such as minimization of part movement, tool changeovers along with the measures of flexibility pertaining to machines, material handling etc. Therefore, the proposed GA-based heuristic has proved to be a promising approach to solving the machine loading problems of FMS by formulating several objective functions, keeping in mind the system constraints.

**Acknowledgements**

The authors have been benefited immensely from discussions with Professor S. K. Mukhopadhyay of the National Institute of Industrial Engineering (NITIE), Bombay, India and Professor S. P. Dutta of the University of Windsor, Canada. They were the source of inspiration and suggested to the authors the complexities of the problem. The authors are also thankful to the several leading researchers, such as Professor K. E. Stecke, Professor Andrew Kusiak, Professor Y. Narahari and Professor Hiroshi Ohta, for extending assistance in the form of sending their valuable reprints.

## References


AMMONS, J. C., LOFGREN, C. B., and MCGINNIS, L. F., 1985, A large scale machine loading problem in flexible assembly, *Annals of Operation Research,* **3**, 319–322.

ANDERSON, E. J., and FERRIS, M. C., 1994, Genetic algorithms for combinatorial optimisation: assembly line balancing problem, *ORSA Journal of Computing,* **6**, 161–173.

BACK, T., 1994, Selective pressure in evolutionary algorithms: a characterisation of selection mechanisms, *Proceedings of the first IEEE Conference on Evolutionary Computation,* pp. 57–62.

BANDYOPADHYAY, S., MURTHY, C. A., and PAL, S. K., 1995, Pattern classification using genetic algorithms, *Pattern Recognition Letters,* **16**, 801–808.

BANDYOPADHYAY, S., and PAL, S. K., 1998, Incorporating chromosome differentiation in genetic algorithms, *Information Science,* **104**, 293–319.

BASNET, C., and MIZE, J. E., 1994, Scheduling and control of flexible manufacturing system: a critical review, *International Journal of Computer Integrated Manufacturing,* **7**(6), 340–355.

BOWDEN, JR., R. O., 1992, Genetic algorithm based machine learning applied to dynamic routing of discrete parts. Ph.D. Dissertation, Department of Industrial Engineering, Mississippi State University, USA.

BUZACOOT, J. A., and YAO, D. D., 1980 Flexible Manufacturing Systems: a review of models: Working paper No. 82-007, University of Toronto, Canada.

CHEN, Y. J., and ASKIN, R. G., 1990, A multiobjective evaluation of flexible manufacturing system loading heuristic, *International Journal of Production Research,* **28**, 895–911.

CHEN, C. L., VEMPATTI, V. S., and ALJABER, N., 1995, An application of genetic algorithm for flowshop problem, *European Journal of Operational Research,* **80**, 385–396.

CLEVLAND, G. A., and SMITH, S. F., 1989, Using genetic algorithms to scheduling flowshop releases, *Proceedings of Third International Conference on Genetic Algorithm and Their Applications* (Palo Alto, CA: Morgan Kaufmann), pp. 160–169.

COX, L. A., DAVIS, L., and QIU, Y., 1991, Dynamic anticipatory routing in circuit-switched telecommunications networks. In *Handbook of Genetic Algorithms* (New York: Van Nostrand Reinhold).

DAVIS, L., 1985a, Job shop scheduling with genetic algorithms. In *Proceedings of the International Conference on Genetic Algorithms and Their Applications,* Carnegie Mellon University, Pittsburgh, PA, pp. 136-140; 1985b, Applying adaptive algorithms to epistatic domains. In *Proceedings of the International Joint Conference on Artificial Intelligence,* pp. 162–164; 1991, *Handbook of Genetic Algorithms* (New York: Van Nostrand Reinhold).

DEJONG, K. A., 1975, Analysis of the behavior of the class of genetic adaptive system. Ph.D. Thesis, University of Michigan Ann Arbor, MI.

FOURMANN, M. P., 1985, Comparison of symbolic layout using genetic algorithm, *Proceedings of the First International Conference on Genetic Algorithms and their Applications* (Hillsdale, NJ: Lawrence Erlbaum Associate), pp. 141–155.

GOLDBERG, D. E., 1987a, Computer aided gas pipeline operation using genetic algorithms and rule learning, Part 1: Genetic Algorithms in Pipeline optimisation, *Engineering with Computers,* 35–45; 1987b, Computer aided gas pipeline operation using genetic algorithms and rule learning, Part 2: Rule learning control of a pipeline under normal and abnormal conditions. *Engineering with Computers,* 47–58; 1989, *Genetic Algorithms in Search, Optimisation and Machine Learning* (Reading, MA: Addison-Wesley).

GOLDBERG, D. E., and LINGLE, R., 1985, Alleles, loci and the travelling salesman problem. In *Proceedings of the First International Conference on Genetic Algorithms and their Applications* (Hillsdale, NJ: Lawrence Erlbaum Associates), **30**(4), pp. 931–940.

GOLDBERG, D. E., KORB, B., and DEB, K., 1989, *Messy Genetic Algorithms: Motivation Analysis and First Results.* Complex System, Complex System Publications Inc., 493–530.

GREFENSTETTE, J. J., 1986, Optimisation of control parameters for genetic algorithms, *IEEE Transactions of Man, Machine and Cybernetics,* **16**(1), 122–128; 1989, A system for learning control strategies with genetic algorithm, *Proceedings of the Third International Conference on Genetic Algorithm* (Palo Alto, CA: Morgan Kaufmann).

GREFENSTETTE, J. J., GOPAL, R., RORMATIA, B., and VANGUCHT, D., 1985, Genetic algorithms for travelling salesman problem In *Proceedings of the First International Conference on Genetic Algorithms and their Applications* (Hillsdale, NJ: Lawrence Erlbaum Associates).

GUPTA, M., GUPTA, Y., and EVANS, G., 1993a, Operations planning and scheduling problems, *International Journal of Production Research,* **31**(4), 869–900.

GUPTA, M., GUPTA, Y., and KUMAR, A., 1993b, Minimising flowtime variance in single machine system using genetic algorithm, *European Journal of Operational Research,* **70**, 289–303.

GUPTA, M., GUPTA, Y., KUMAR, A., and SUNDRAM, C., 1996, A genetic algorithm-based approach to cell composition and layout design problems, *International Journal of Production Research,* **34**(2) 447–482.

HERDY, M., 1991, Application of the evolution strategy to discrete optimisation problems. *Proceedings of the First International Conference on Solving Parallel Problems Solving from Nature.* Lecture Notes on Computer Science, 496 (Springer Verlag), pp. 188–192.

HERRMANN, J.-W., and LEE, C.-Y., 1995, Solving a class scheduling problem with a genetic algorithm, *ORSA Journal of Computing,* **7**, 443–452.

HERRMANN, J.-W., LEE, C.-Y., and HINCHMAN, J., 1995, Global job shop scheduling with a genetic algorithm, *Productions and Operations Management,* pp. 430–444.

HOLLAND, H. H., 1975, *Adaptation in Natural and Artificial Systems* (Detroit MI: University of Michigan Press).

HWANG, S. S., and SHOGUN, A. W., 1989, Modeling and solving FMS part selection problem, *International Journal of Production Research,* **27**(8), 1349–1366.

JAIN, A. K., and ELMARAGHY, H. A., 1997, Production scheduling/rescheduling in flexible manufacturing, *International Journal of Production Research,* **35**(1), 281–309.

JAIN, S., BARBER, K., and OSTERFLED, D., 1989, Expert simulation for on-line scheduling, *Proceedings of the 1989 Winter Simulation Conference,* pp. 930–935.

KIM, Y. D., and YANO, C. A., 1997, Impact of throughput based objectives and machine grouping decisions on the short-term performance of flexible manufacturing systems, *International Journal of Production Research,* **35**(2), 3303–3322.

KUMAR, P., SINGH, N., and TIWARI, N. K., 1987, Multicriterion analysis of the loading problem in flexible manufacturing system using min-max approach, *International Journal of Advanced Manufacturing Technology,* **2**(2), 13–23.

KUSIAK, A., 1984, The part families problem in flexible manufacturing system. *Proceedings of the First ORSA/TIMS Conference on FMS* (Ann Arbor, MI: University of Michigan), pp. 237–242; 1985, Loading models in flexible manufacturing systems. In *Manufacturing Research and Technology-1,* edited by A. Raouf and S. H. Ahmed (Amsterdam: Elsevier).

LASKARI, R. S., DUTTA, S. P., and PADHE, A. M., 1987, A new formulation of operation allocation problem in flexible manufacturing system: mathematical modeling and computational experience, *International Journal of Production Research,* **25**(9), 1267–1283.

LEE, C.-Y., PIRAMUTHU, S., and TSAI, Y.-K., 1997, Job shop scheduling with genetic algorithm and machine learning, *International Journal of Production Research,* **35**(4), 1171–1191.

LIANG, M., and DUTTA, S. P., 1992, Combined part-selection, load sharing and machine-loading problem in hybrid manufacturing system, *International Journal of Production Research,* **30**(10), 2335–2350; 1993, An integrated approach to part-selection and machine-loading problem in a class of flexible manufacturing systems, *European Journal of Operational Research,* **67**(3), 387–404.

LIEPINS, G. E., and HILLIARD, M. R., 1989, Genetic algorithms: foundations and applications, *Annals of Operations Research,* **21**, 31–58.

MICHALEWICZ, Z., 1992, *Genetic Algorithm + Data Structure = Evolution Programs* (New York: Springer Verlag).

MORENO, A. A., and DING, F.-Y., 1993, Heuristics for the FMS loading and part type selection problems, *International Journal of Flexible Manufacturing Systems,* **5**, 287–300.

MUKHOPADHYAY, S. K., and TIWARI, M. K., 1995, Solving machine loading problems of FMS using conjoint measurement, *Proceedings of 13th International Conference on Production Research,* Jerusalem, pp. 74–76.

MUKHOPADHYAY, S. K., MAITI, B., and GARG, S., 1991 Heuristic solution to the scheduling problems in flexible manufacturing systems, *International Journal of Production Research,* **29**(10), 2003–2024.

MUKHOPADHYAY, S. K., MIFHA, S., and KRISNA, V. A., 1992, A heuristic procedure for loading problems in FMSs, *International Journal of Production Research,* **30**(9), 2213–2228.

OLIVER, I. M., SMITH, D. J., and HOLLAND, J. R. C., 1987, A study of permutation crossover operators on the travelling salesman problem. In *Proceedings of the Second International Conference on Genetic Algorithms* (Hillsdale, NJ: Lawrence Erlbaum Associates), pp. 224–230.

PAKATH, R., and ZAVERI, Z. S., 1993, Specifying critical inputs in a genetic driven decision support system: an automated facility. Working paper, University of Kentucky, Lexington.

RAJGOPALAN, S., 1986, Formulation and heuristic solutions for part grouping and tool loading in flexible manufacturing systems, *Proceedings of the 2nd ORSA/TIMS Conference on FMS,* University of Michigan, Ann Arbor, Michigan, USA, Vol. 5, pp. 312–320.

RAM, B., SARIN, S. C., and CHEN, C. S., 1990, A model and solution approach for machine loading and tool allocation problem in a flexible manufacturing system, *International Journal of Production Research,* **28**(4), 637–645.

SAWAKA, M., KATO, K., and MURI, T., 1996, Flexible scheduling in a machining center through genetic algorithm, *Computers and Industrial Engineering,* **30**(4), 931–940.

SARIN, S. C., and CHEN, C. S., 1987, The machine loading and tool allocation problem in a flexible manufacturing system, *International Journal of Production Research,* **25**(7), 1081–1094.

SCHAFFER, J. D., CARUANA, R. A., ESHELMAN, L. J., and DAS, R., 1989, A study of control parameters affecting on-line performance of genetic algorithms for function optimisation. In *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications* (Palo Alto, CA: Morgan Kaufmann Publishers), pp. 51–60.

SHNAKAR, K., and SRINIVSULU, A., 1989, Some solution methodologies for loading problems in flexible manufacturing system, *International Journal of Production Research,* **27**(6), 1019–1034.

SHANKER, K., and TZEN, Y. J., 1985, A loading and dispatching problem in a random flexible manufacturing system, *International Journal of Production Research,* **23**(3), 579–595.

SINGHAL, K., 1978, Integrating production decisions, *International Journal of Production Research,* **16**(5), 383–393.

STARKWATHER, T., MCDANIEL, S., MATHIAS, K., WHITLEY, C., and WHITLEY, D., 1991, A comparison of genetic sequencing operators, *Proceedings of the 4th International Conference on Genetic Algorithms* (Los Altos CA: Morgan Kaufmann Publishers), pp. 69–76.

STECKE, K. E., 1983, Formulation and solution of nonlinear integer production planning problem for flexible manufacturing system, *Management Science,* **29**(3), 273–288; 1985, A hierarchical approach to solving grouping and loading problems of flexible manufacturing systems, *European Journal of Operational Research,* **24**(3), 369–378.

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.

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.

VAN LOOVEREN, A. J., GELDERS, J. L. F., and VAN WASSENHOVE, L. N., 1986, A review of FMS planning models. In *Modeling and Design of Flexible Manufacturing Systems,* edited by A. Kusiak (Amsterdam: Elsevier), pp. 3–31.

WHITNEY, C. K., and GAUL, T. S., 1984, Sequencial decision procedure for batching and balancing in FMSs, *Proceedings of the First ORSA/TIMS Conference on FMS.* University of Michigan, Ann Arbor, Michigan, USA, pp. 243–248.