A Genetic Algorithm Based Application for a Flexible System

S. Wadhwa, J. Madaan, and R. Raina

Indian Institute of Technology—Delhi (IIT-D),

Hauz Khas, New Delhi, India

Abstract: Based on the rapid technical development in the last decade, the automation in many areas of production & distribution has developed significantly. This leads to complex situations where decisions have to be taken within a short time and among several alternatives – often without the human intervention. This paper considers flexible system as a mixed model flexible flow line, where 'n' independent jobs are required to be processed on 'm' different machines, where all the jobs have the same processing order on the machines. Here the objective is to find the ordering of the jobs on the machines that minimizes the make-span. Further, it proposes the use of evolutionary based heuristic of genetic algorithm on the flow line scheduling problem. Proposed architecture has been developed to depict the application of genetic algorithms to optimize production schedules in a flexible flow line system representing a flexible system. Results show that the implementation of the genetic algorithm is very effective as compared to standard sequencing rules like shortest processing time, total processing time, etc. and at the same time easy to use. Finally this paper intends to discuss some of these interesting results with a focus on application of lead-time reduction in an interesting flexible system like RES (Reverse Enterprise System).

Keywords: Flexible System, Genetic Algorithm, Flexible Flow Line, Simulator, RES

1. Introduction

The rapid advancement of technology is resulting in shortened life cycles and putting great pressure on organizations to achieve shorter and shorter manufacturing lead times. Hence, one of the important focuses of manufacturing enterprises is to find ways and means of reducing the manufacturing lead times to be able to respond quickly for changing market demands. Towards this end, scheduling has always been considered as one of the most important issues for lead time reduction in a mixed model flexible system. Due to this complexity of scheduling problems an efficient and effective advanced scheduling approaches are in great demand. Considering a mixed model flow shop scheduling problem, where 'n' independent jobs required to be processed on 'm' different machines, as a case representing a flexible system. Flow shop scheduling has been proved to be NP-hard in strong sense [Garey et. al, 1979]. Till now, mathematical programming, constructive heuristics and meta-heuristics have been proposed for a generic flow shop scheduling [Reeves, 1995; Ogbu, 1990; Wang et. al, 2003, Chan et. al 2005], where it is usually assumed that the buffer size between every two successive machines is infinite. However, in real practice, such as the petrochemical processing industries, reverse logistics system and electronic chip manufacturing system, buffer is non-existent. In such case, a job is called blocked if its operation on a certain machine has finished but the machine after that is still busy.

A number of knowledge-based system (KBS) techniques for reconfiguring flexible systems have been proposed in the literature. Wadhwa & Brown (1989) had proposed potential of basic Petri net concepts for graphic representation, specification purposes, and control of flow in a flexible system. Since these KBS technique rely heavily on past experience. The required information sometimes may not be available or difficult to be obtained. Hence genetic algorithm (GA) is well a suited tool to tackle such complex reconfiguration problems, because GA uses information-randomized (genes) exchange to exploit to tackle complex scheduling problems.

Rapid evolution in computer technology has shown advancement in the field of artificial intelligence (AI) and some of the AI techniques have been applied to the business in recent years [Grupe et. al, 2004]. Since the initial conception by John Holland in 1975, there has been a growing interest in Genetic Algorithms (GAs) – optimization algorithms based on the principles of natural evolution. Based on the mechanics of artificial selection and genetics, GAs are population-based evolutionary searching techniques, which combine the concept of survival of the fittest among solutions with a structured yet randomized information exchange and offspring creation [Goldberg, 1989]. They have been used widely for parameter optimization, classification and learning in a wide range of applications. Recently, production scheduling has emerged as an application. This paper deals with finding of an ordering of the jobs on the machines or sequences that optimizes the make-span. The area of flow-shop scheduling has been a very active field of research during the last 50 years since Johnson's seminal work in 1954. The desire to seek the perfect solution is driven by the high dependence of a manufacturer's performance and profitability on schedule optimization. Multiple implications of the results obtained from the present study can made towards developing schedule for lead-time reduction in flexible systems.

This paper presents a genetic algorithm based architecture which has been developed to find the best sequence for scheduling the product orders with the objective of minimizing the make-span of the whole process. Its applicability has been demonstrated by the means of a simulator developed as a part of this paper. The simulator is developed in Java and incorporates the basic principles of genetic algorithm like crossover and mutation along with modifications like elitism. The effect of varying the basic parameters of the genetic algorithm on the running of simulation for calculating make-span has also been presented.

2. What is Flexible System?

Several attempts have been made in the literature to define, model, and measure the performance of the flexible system. Many definitions of a flexible system can be found in the literature. For instance, Carlsson (1989) cite a flexible system as a system which accommodates greater output variation, as the firm's response to uncertainty, especially in the form of fluctuations in demand, but also market imperfections. Benjaafar and Ramakrishnan (1996) cite the following definition from literature the capacity of a system to assume different positions or to assume a certain number of different states. Wadhwa and Rao (2000) view flexible system as system that has ability to deal with change by judiciously providing and exploiting controllable options dynamically. Further, Wadhwa and Rao (2003) use many flexibility concepts to discuss multiple entity flows from an enterprise synchronization perspective in the context of supply chains. There is a growing work on flexibility in manufacturing and supply chain systems that may be viewed as flexible systems (Wadhwa and Rao, 2004). However there is need to promote research in flexibility in reverse enterprise systems (RES) also. In our opinion the flexibility needs in reverse enterprise systems are higher than most conventional systems and hence RES also need to be treated as flexible systems. For instance, a flexible scheduling for product return may be devised, which can ensure a reduction in lead time among the closed loop supply-chain partners (nodes). This will reduce the time required for product differentiation, cost of time delays, and its effect on asset recovery in a Reverse Enterprise System (Wadhwa & Madaan, 2004) as a whole. Our ongoing research (e.g. Wadhwa and Madaan, 2006) reflects good possibilities in this direction. In this paper we first focus on flexible systems. Then we consider the case of flexible flow lines and show how GA applications can help improve the performance of these flexible systems. The performance is compared between GA based approach and other strategies. Finally we outline how a similar approach may also be applied to reverse enterprise systems viewed as flexible systems.

3. Motivation for Study

With respect to its modeling potential, GA can model a wide spectrum of manufacturing systems scenarios. This tool can be very effective in cases with a large variety of jobs/products following the same sequence on a large number of machines/system nodes. The flexible flow-shop scheduling problem we consider here is known to be NP-complete (non-deterministic polynomial) in the strong sense when number of machines, $m \geq 3$ [Garey et. al. 1976]. If $m = 2$ Johnson's algorithm obtains an optimal solution in polynomial time. Being NP-hard, there are no mathematical techniques yet available to solve these problems for optimality. For a simple n jobs and m machines schedule, the upper bound on the number of solutions is $(n!)^m$. This paper deals with the simplification of FSP, which is the Permutation Flow-shop Scheduling Problem (PFSP). Here in the PFSP, job passing is not permitted, i.e. the processing sequence of jobs is the same for all machines. Under this consideration, '$n!$' schedules are possible. Traditional approaches to schedule optimization are computationally slow in such enormous search space. Thus, brute force methods are not an option in this situation.

Moreover, traditional sequencing rules provided a limited number of solutions which were rarely optimal. Therefore need to develop an algorithm to find the best sequence for scheduling the product orders with the objective of minimizing the make-span of all the jobs is there. Hence GA is a tool used for solving combinatorial and optimization problems [Grupe et. al, 2004]. In cases with large search space, genetic algorithm is effective in providing solutions close to optimal. Thus GA is computationally fast as it directionally searches a limited amount of feasible solutions rather than searching all solutions simultaneously.

4. Literature Review

The area of flow-shop scheduling has been an active field of research during the past few decades since Johnson's seminal work back in 1954. Many researchers have used genetic algorithms for solving scheduling problems. The effectiveness of GA is enhanced by the fact that traditional scheduling problems are considered to be NP-hard and therefore they cannot be solved for the optimal solution in finite amount of time [Wadhwa & Chopra, 2000].

Wang et al. (2006) in their work propose an effective hybrid GA model for flow-shop scheduling with limited buffer. Here local search based on graph model is employed and multiple crossover and mutation operators are used simultaneously, so that exploration & exploitation abilities can be well balanced. The effects of buffer size & decision probability on optimization quality are discussed.

Wadhwa et al. (2005) indicates the existence of complex interactions between the sequencing flexibility, process concurrency, processor load balancing, and manufacturing lead time. This paper intends to discuss some of the interesting results with a focus on the inherent mechanisms of sequencing flexibility-enabled lead-time reduction.

Further Sadegheih (2006) displays the use of GA to optimize production schedules. According to them, the advantage of GA is that it provides a general purpose solution to the scheduling problem, with the peculiarities of any particular scenario being accounted for in fitness function without disturbing the logic of the standard optimization routine. They also show that overall Simulated Annealing (SA) needed longer computation time than GA.

Ruiz et al. (2006) proposed new GA for a generalized permutation flow-shop scheduling problem. Extensive comparison with other algorithms showed that the proposed GA outperforms all other algorithms. Their problem has a number of unrelated parallel machines at each stage, sequence dependent setup times, and machine eligibility. The statistically results indicate that the proposed algorithm is between 53% & 135% better than the second best method.

Grupe and Jooste (2004) in their work describe GA as being applied to solve business problems. There exists a cardinal rule of computing, "use the right tool for the right problem." They present the criteria for the application of GA. They also point out the limitations of GA, the major ones being that not all problems can be framed in the mathematical manner that GA demands. Moreover, development of a GA and interpretation of the results requires an expert who has both programming and statistical/mathematical skills demanded by the GA technology in use.

Silva et al. (2005) in their work discuss the methodologies that can be used to optimize a logistic process of a supply chain described as a scheduling problem which are NP-hard. They introduced the concept of cross-docking centers which are facilities where goods can be deposited for short periods of time before being picked up. Here a new objective function called Global Expected Lateness is proposed, in order to describe multiple performance improvement criteria. This function minimizes the sum of the lateness absolute values of all delivered orders. Based on the optimization methodologies proposed, the results show that dispatching heuristics are outperformed by the GA meta-heuristics.

It is useful to apply GA in the flexible systems. The reason is that these systems are very complex and involve intricate control strategies to exploit flexibility. For instance Wadhwa and Rao (2006) compare the performance of three flexibility types in the manufacturing context. Chan et al. (2004) have also discussed flexible systems and their inherent complexity. Wadhwa and Rao (2000) have discussed various types of flexibility and described the underlying complexity to deal with such systems. It is proposed that GA can be used in many of these systems to improve their performance. The GA approach can also help to improve our knowledge to manage flexible systems more effectively. Wadhwa and Madaan (2006) have shown how knowledge management can help to improve RES viewed as flexible systems. Klassen & Angel (1998) illustrated a comparative study for measuring the impact of manufacturing flexibility in environmental management systems. Hokey et al. (2006) have shown how nonlinear mixed-integer programming model and GA may be applied to RES systems for performance benefits. However there is a need to show how RES may be seen as flexible systems and thereby more results and experiences can be translated and used in RES viewed as flexible systems.

Hurink (2005) defines the basic structure of local search and moves on to iterative improvement of local search and then to simulated annealing, tabu search and genetic algorithm. He considers the problems with minimization of objective function as the criterion.

Whitley (2005) covers the canonical genetic algorithm as well as more experimental forms of genetic algorithms including parallel island models and parallel cellular genetic algorithms. He also illustrates genetic search by hyper-plane sampling. The theoretical foundations of genetic algorithms are reviewed, include the schema theorem, as well as recently developed exact models of the canonical genetic algorithm.

In this paper we use a mix model flexible line to show the usefulness of GA applications. More complex flexible systems such as FMS and RES will have even a greater need for GA based approaches. The approach presented here may be extended to other flexible systems also. A simulation model of a sample flexible flow shop is used to show the benefits of GA.

5. Sample Flexible System Model

A sample flexible flow-shop may be viewed to comprise of a set N = $\{J_1, J_2... J_n\}$ of *n* independent jobs that have to be processed on a set M = $\{M_1, M_2... M_n\}$of *m* different machines. Each job requires a known, deterministic, and non-negative span of time for completion at each flexible machine. The machines are flexible as they can process different job types. The machine flexibility (Browne et al 1984) here reflects the changeover time at a machine to process a changed job. The operation time of each job at each machine is different. This time is called processing time and it is denoted as p_{ij} ($i = 1\ldots n, j = 1\ldots m$}. In a flow-shop, all jobs have to be processed sequentially on the machines and have the same processing order. The objective is to find an ordering of the jobs on the machines that optimizes some given criterion. The criterion used in this project is the minimization of the total completion time of the schedule, often referred to as make-span, $C_{max.}$ The problem is classified as n/m/F/C_{max} [Conway et. al, 1967] or the three parameter notation (α/β/γ) can be followed [Graham et. al, 1979]. α indicates the state of the problem, which in out case is the flow-shop, i.e. F. β field cumulates a set of explicit constraints. In case of permutation flow-shop scheduling, β is replaced by *permu.* γ field indicates the objective to be optimized. Hence our problem can be represented as F/permu/C_{max}. This scope of this work is limited only to permutation schedules, i.e. job passing is not permitted, and therefore, the processing sequence of the jobs is same for all the machines.

It is important to note several additional conditions to the problem. All operations are independent and available for processing at time zero. All m machines are continuously available. Each machine *i* can process at most one job *j* at a time. Each job *j* can be processed on at most one machine *i* at a time. The processing of a given operation cannot be interrupted, i.e. no preemption is allowed. Setup or removal times are sequence independent and are included in the processing times or otherwise are negligible and can be ignored. If a job needs a machine downstream which is unavailable, it has to wait at its current position till the machine becomes available, i.e. no buffers have been provided into the system. A simulation model for the flexible flow shop is developed. This is integrated with a GA based scheduling system to help improve the make-span performance of the flexible system.

6. The GA Application Model

The GA based application model of the flexible system starts with an initial population which is obtained by use of standard sequencing rules along with some random chromosomes. Each chromosome is then evaluated by calculating the value of make-span. This value of make-span is used to calculate the fitness value of each chromosome. Fitness value is inversely proportional to the make-span. Based on this fitness value, we start the process of selection for the next generation. The chromosome with the best fitness value is directly placed in the new population. This step is called elitism and it prevents the loss of the best solution because of evolutionary techniques. The remaining chromosomes for the next generation are selected on the basis of roulette wheel selection. Each chromosome is given a section of roulette wheel based on its fitness value as explained in last chapter. Then random numbers are generated and the chromosome corresponding to the region in which the random number falls is selected. After selection, it is the time to apply the crossover and mutation operators on the intermediate population. Once these operations are complete, the new population is ready to be evaluated again. This process repeats till a specified number of iterations.

Algorithm for the GA has been demonstrated below.

[Start] Generate random population of *n* chromosomes (suitable solutions for the problem)

[Fitness] Evaluate the fitness *f(x)* of each chromosome *x* in the population

[New population] Create a new population by repeating following steps until the new population is complete.

[Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)

[Crossover] With a crossover probability, cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.

[Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).

[Accepting] Place new offspring in the new population.

[Replace] Use new generated population for a further run of the algorithm.

[Test] If end condition is satisfied, stop, and return the best solution in current population.

[Loop] Go to step [new population].

The mathematical model for make-span calculation is as follows. Let ($\pi_1, \pi_2, \pi_3 \ldots \pi_n$) be a permutation. Computing the finish time $C(\pi_i, j)$ for the ith job of the given permutation π and the machine j, can be done as follows:

$$\left.\begin{array}{ll}
C(\pi_1, 1) = p_{\pi_1,1} & \\
C(\pi_i, 1) = C(\pi_{i-1}, 1) + p_{\pi_i,1} & i = 2, \ldots, N \\
C(\pi_1, j) = C(\pi_1, j-1) + p_{\pi_1, j} & j = 2, \ldots, M \\
C(\pi_i, j) = \max\{C(\pi_{i-1}, j), C(\pi_i, j-1)\} + p_{\pi_i, j} & i = 2, \ldots, N; j = 2, \ldots, M
\end{array}\right\}$$

Under these specifications, the value of the objective function, the make-span, C_{max}, is given by $C(\pi_N, M)$ – completion time for the last operation on the last machine.

Make-span of the PFSP, $C_{max} = C(\pi_N, M)$

The value of the fitness function of chromosome *i* is given by *F (i)*, where

F(i) = (Minimum make-span in the population) / (Make-span of chromosome *i*)

The section on the roulette wheel occupied by chromosome i is given by R (i), where

$R(i) = \{ F(i) / \Sigma F(i) \} * 360$

7. A GA Based Architecture

To solve the PFSP, an overall approach has been developed and cases in literature related to GA application in other domains have been studied and analyzed. This paper conducts an in depth study of the GA algorithm has been carried out to understand overall functions. An overall architecture for GA based optimization of a flow-shop scheduler model has been developed as shown in Figure 6.1. It has been designed using a modular approach. The key modules identified are scheduler model, which provides the basic input population and the fitness function for the particular problem; the extractor, which extracts the input data from the scheduler; GA module which implements GA techniques like selection, crossover, and mutation to the data and the checker module which checks whether the result obtained is optimal. If the solution is optimal, it is sent as output; else it is sent to the changer module for generating a new population. These modules suitably interact with scheduling environment and with each other in order to fulfill the functionality of GA directed improvement process. Based on the architecture designed, software codes have been developed to test the efficacy of the approach. These codes provide a simple and robust method to apply GA to the flow-shop scheduling problem. The code has been written in Java and also provides a user friendly Graphical User Interface as shown in Figure 6.2.

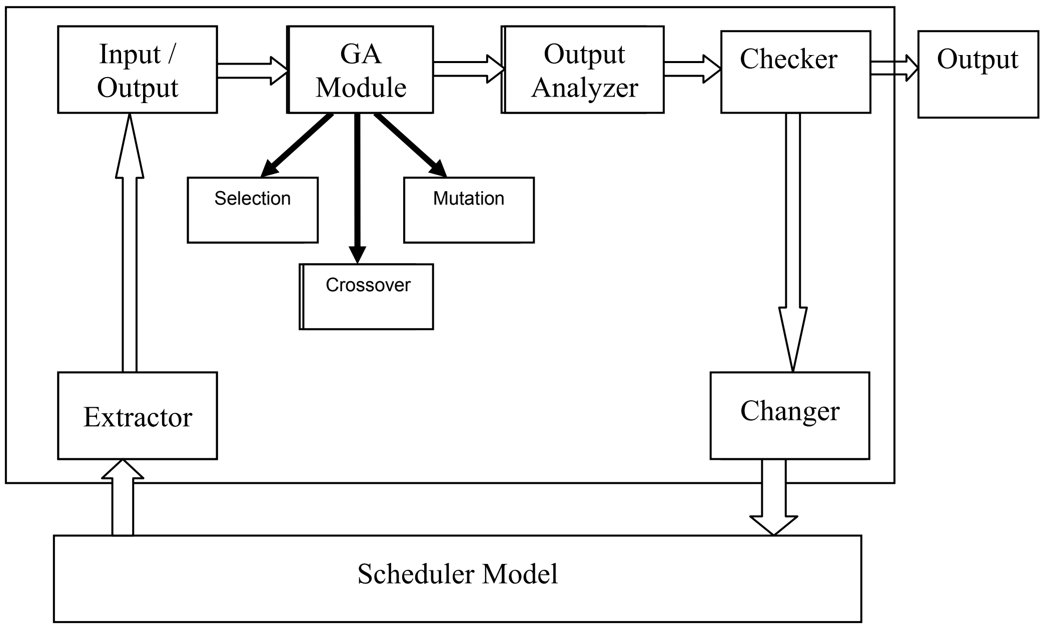
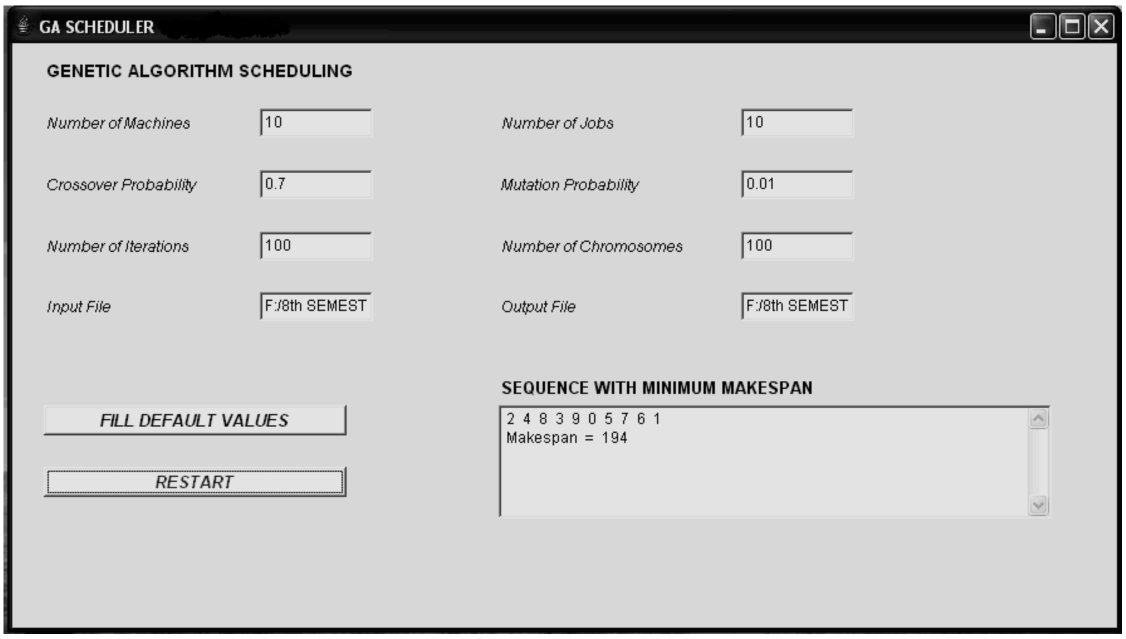


Figure 6.1. Architecture of GA in Detail as Applied to Scheduler Model


GA SCHEDULER

GENETIC ALGORITHM SCHEDULING

Number of Machines: 10
Number of Jobs: 10
Crossover Probability: 0.7
Mutation Probability: 0.01
Number of Iterations: 100
Number of Chromosomes: 100
Input File: F:/8th SEMEST
Output File: F:/8th SEMEST

SEQUENCE WITH MINIMUM MAKESPAN

```
2 4 8 3 9 0 5 7 6 1
Makespan = 194
```

Figure 6.2. Graphical User Interface for the GA Scheduler

Here the GUI acts as a link between the end-user and the software code. The input parameters for the GA scheduler are the number of machines and the number of jobs to be processed on the machines. The software has been successfully tested for the problem size as high as 1000 machines and 1000 jobs. The GUI also provides an option to the user to enter the crossover probability and the mutation probability but care must be taken while entering these values as the performance of GA is seriously affected by them. The user must also set the number of iterations for which he wishes to run the software and also the size of the population used. The affects of variation in these parameters have been shown in the next chapter. The GA scheduler also provides an option to the user to load the input values by reading from a file. For this the user must enter the full path of the input file in the GUI. Similarly the output file path must also be entered in the GUI.

The parallel approach of GA has been illustrated in Figure 6.3. It shows that a population sample consists of chromosomes from various regions of the search space rather than being limited to a particular region.

This allows the GA to traverse a large search space and look for optimal solutions without falling into local minima.

The working of the genetic algorithm can be visualized by Figure 6.4. Here, a plot of the best value of make-span during iteration against the iteration number is made. It can be seen that how the genetic algorithm progresses from one generation to another, thereby improving the solution value. The flat regions in the figure are the local minima but GA mostly manages to escape from such positions because of its intrinsic evolutionary features like crossover and mutation.

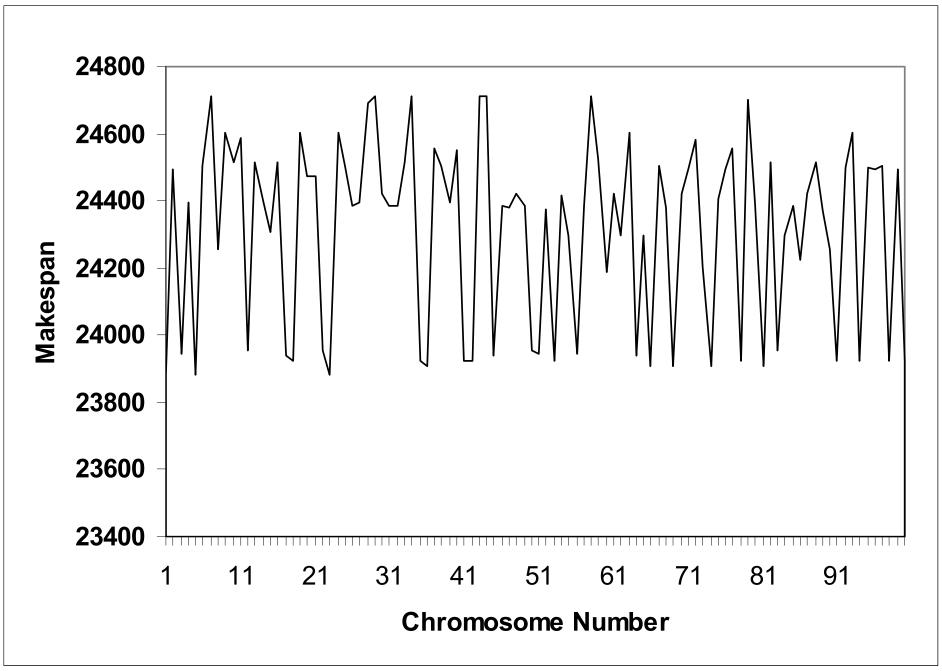


Figure 6.3. Population Sample

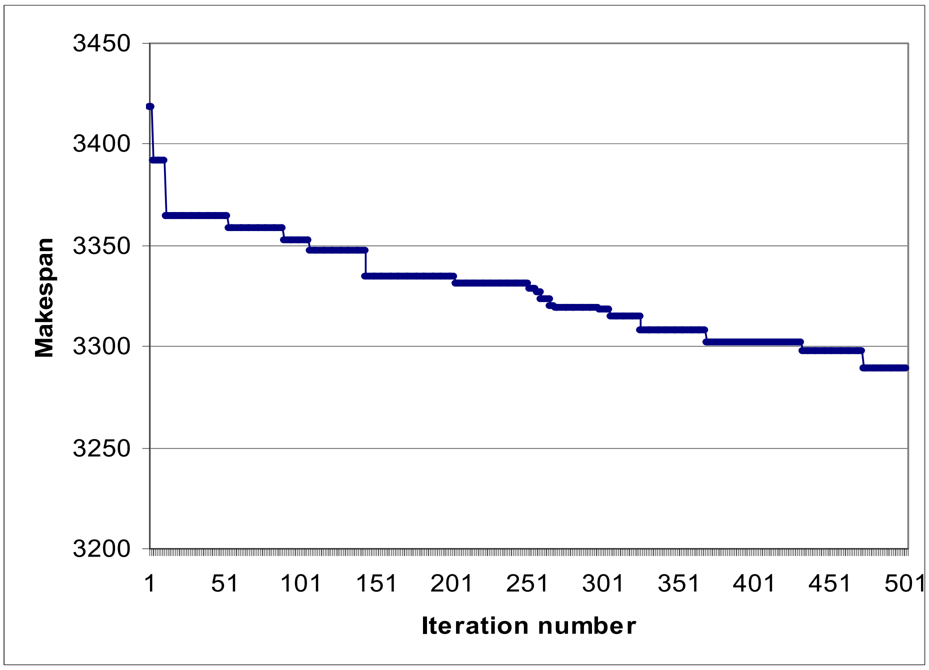


Figure 6.4. Working of GA

7.1. Model Experiments

In this section we compare the results obtained by the use of the GA Scheduler with those obtained by standard sequencing rules like first-in first-out, shortest processing time, total processing time, etc. Figure 6.5 is for a 100 jobs 100 machines case whereas in Figure 6.6 we run the software for a massive search space with 1000 jobs 1000 machines case. As is evident from the inspection of both figures, the performance of the GA Scheduler is far better than the standard sequencing rules, thus demonstrating the efficacy of the approach.

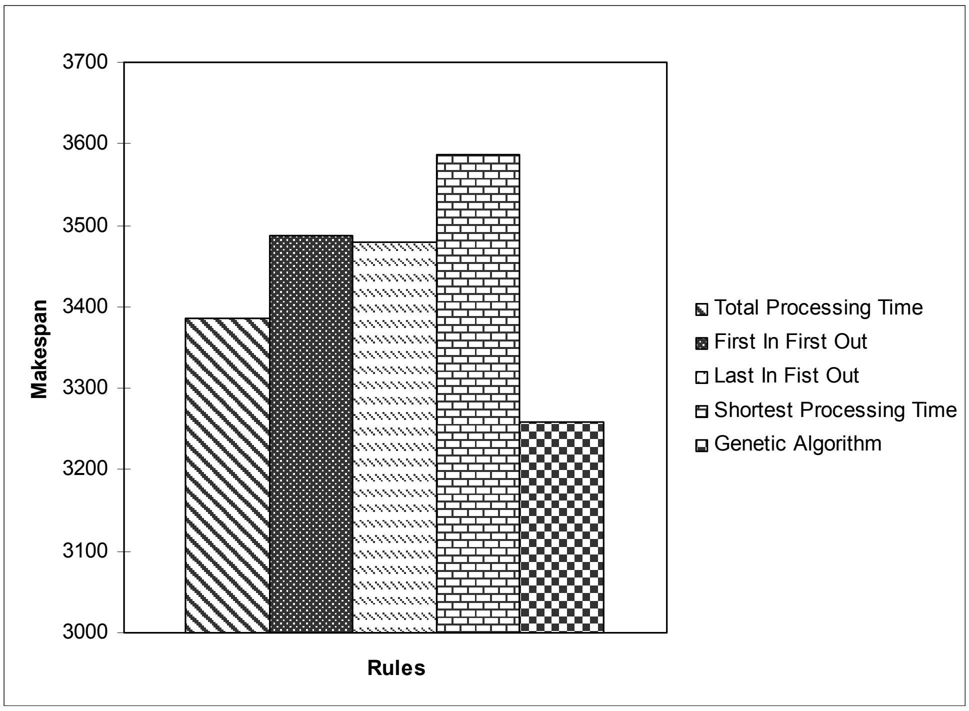


Figure 6.5. Comparison of GA with standard sequencing rules: 100 machines 100 jobs data

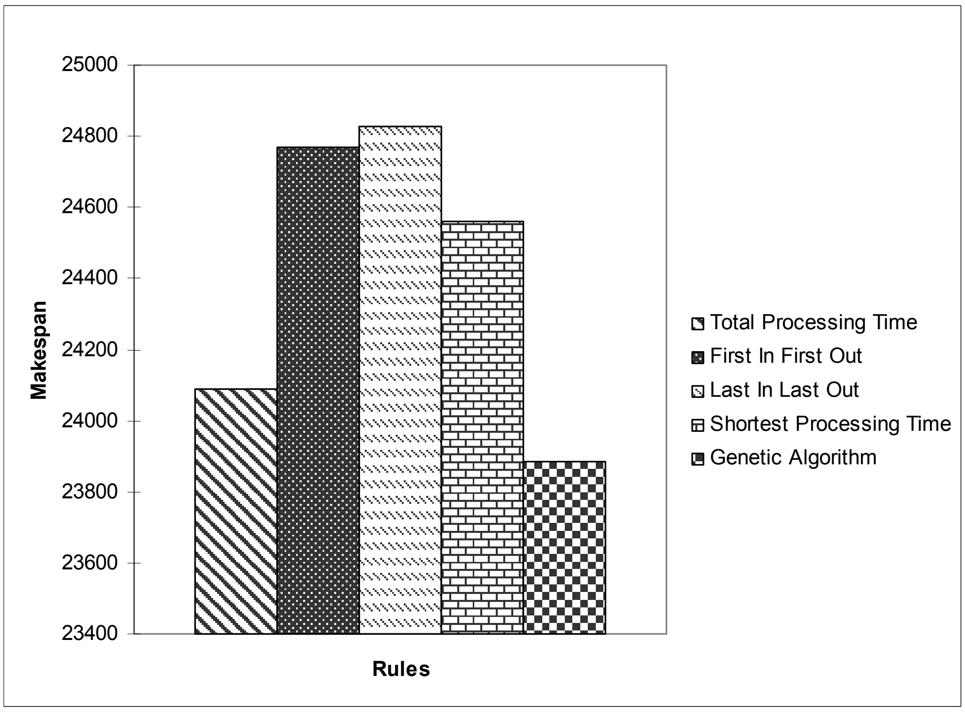


Figure 6.6. Comparison of GA with standard sequencing rules: 1000 machines 1000 jobs data

8. Model Sensitivity Analysis

As expected from an evolutionary technique like genetic algorithm, the next generation always tends to improve the solution. Increasing the number of iterations effectively increases the number of generations or evolutions. As a result, the chromosomes tend to move away from the local optima towards a globally optimal solution. This can be seen in Figure 7.1 where the make-span value keeps on decreasing as we increase the number of iterations. If we increase the number of iterations further, the solution will keep on improving till we reach global optima. Selecting the right number of iterations depends on the problem size and one requires experience to make correct decision.

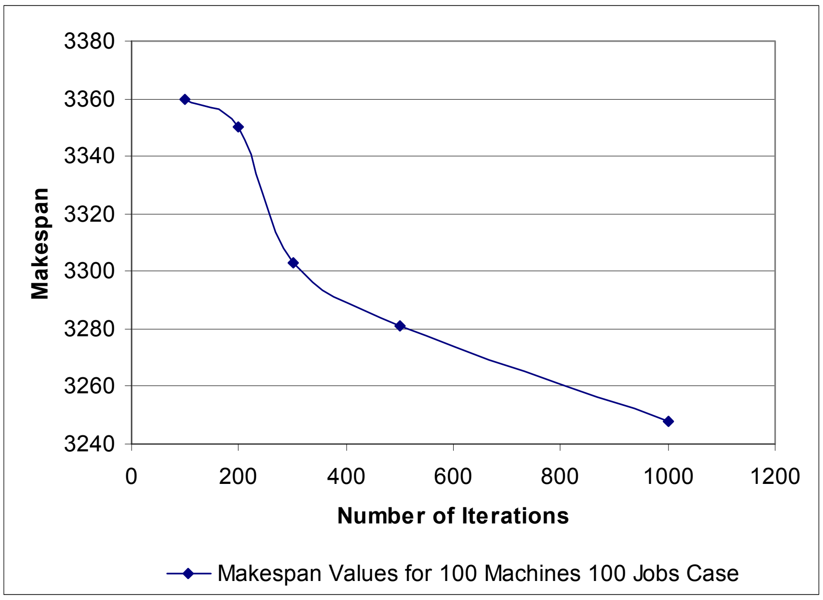


Figure 7.1. Effect of changing the number of iterations

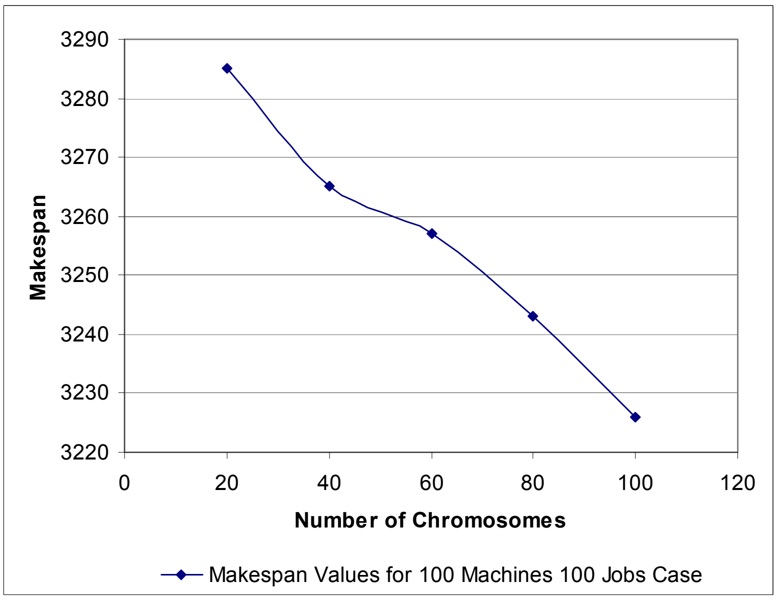


Figure 7.2. Effect of changing the size of the population

Population size is another important factor in GA. It determines how many chromosomes are in population (in one generation). If there are too few chromosomes, GA has few possibilities to perform crossover and only a small part of search space is explored. But having too large a population size can lead to redundancy which eventually slows down the output and reduces efficiency. Research shows that after some limit it is not useful to use very large populations because it does not solve the problem faster than moderate sized populations.

Here Figure 7.2 shows the effect of varying the number of chromosomes on the performance of GA by considering 100 machines 100 jobs case. It shows clearly that having a wider set of chromosomes to choose from results in reaching the optimal value earlier. This can be attributed to the fact that genetic algorithm uses tools like crossover and mutation which can help to arrive at the desired sequence of jobs if we have sequences from different areas of the search space. Selecting the right population size depends on the problem size and one requires experience to make correct decision.

Crossover probability determines how often crossover will be performed. If there is no crossover, offspring are exact copies of parents. If there is crossover, offspring are made from parts of both parent's chromosome. If crossover probability is 100%, then all offspring are made by crossover. If it is 0%, whole new generation is made from exact copies of chromosomes from old population. Crossover is made in hope that new chromosomes will contain good parts of old chromosomes and therefore the new chromosomes will be better. However, it is good to leave some part of old population survives to next generation.

Figure 7.3 shows the effect of changing the crossover probability. As we can see, having crossover probability in the range < 0.6 leads to more number of iterations to reach the optimal solution. Having crossover probability in the range > 0.8 can result in the loss of good chromosomes of the previous population, which leads to slower convergence to the optimal solution. Our results show that the crossover probability should be in the range 0.6 to 0.8 for better working of the GA as documented in literature.

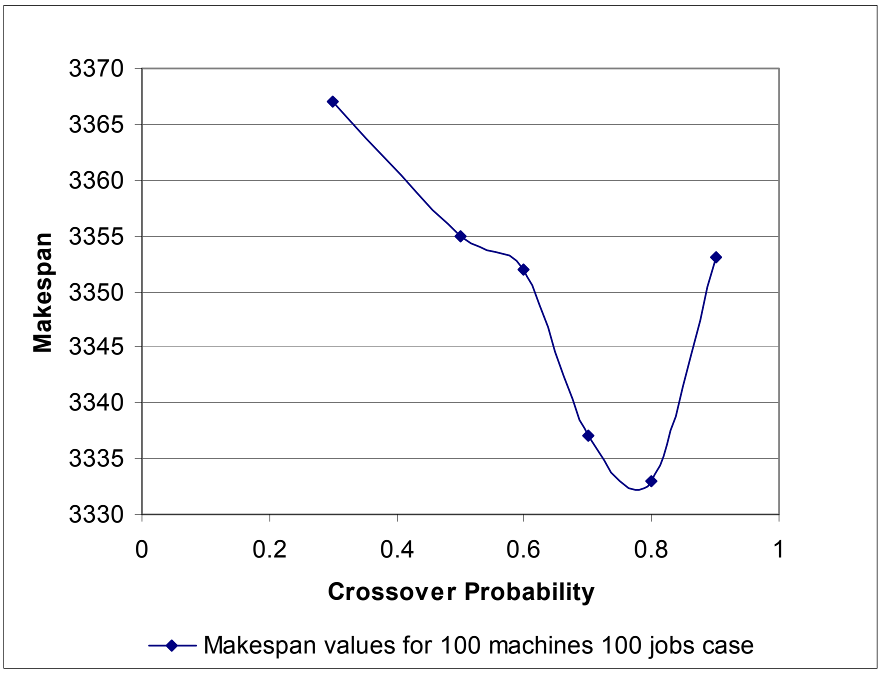


Figure 7.3. Effect of changing the crossover probability

Mutation probability determines how often the chromosome will be mutated. If there is no mutation, offspring are generated immediately after crossover without any change. If mutation is performed, one or more parts of a chromosome are changed. Mutation generally prevents the GA from falling into local extremes. Mutation should not occur very often, because then GA will in fact change to random search.

Figure 7.4 shows the effect of changing the mutation probability on the value of make-span. Our results show that the value of mutation probability which gives good results should be in the range of 1%. This result has been validated even in the past research works.

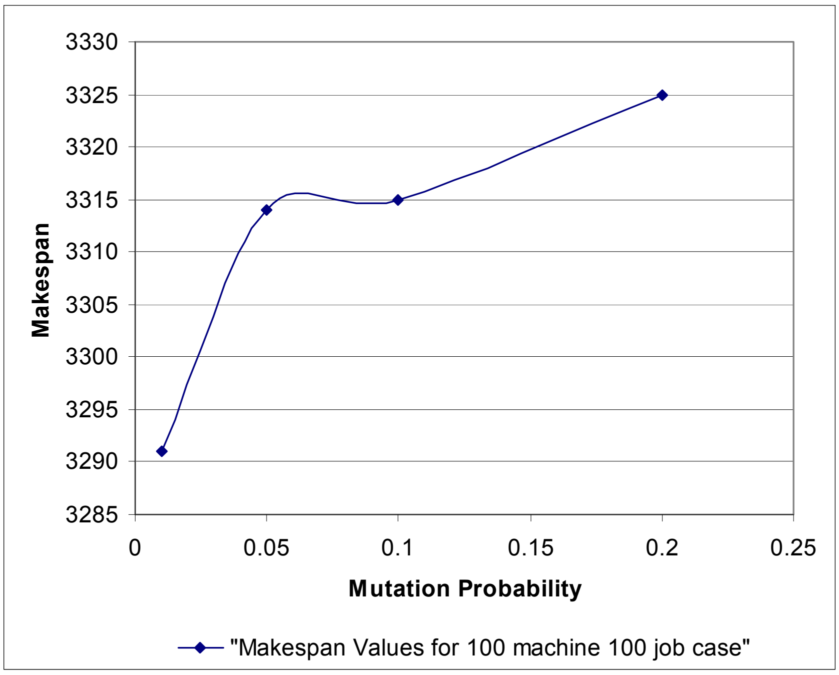


Figure 7.4. Effect of changing mutation probability

9. Comparison of GA with Other Approaches

In the past, many other heuristics have been also successfully applied to scheduling problem like simulated annealing and tabu search. This section compares the use of GA with other algorithms like simulated annealing and tabu search.

In simulated annealing (SA), we successively generate a random move in the neighborhood of the state, accept that move if it leads to a solution with lower energy than the current solution, and otherwise accept it with some positive probability. At each stage, we have to generate a feasible neighboring solution and then decide whether to move from the current solution to the neighboring solution. This makes SA slower as it is confined to a neighborhood of a single solution and moves from one solution to another. On the other hand, GA traverses a search space in a parallel fashion, thus allowing it to look for global optima without being stuck into local optima. This reasoning is confirmed by Wu (2005), who says that genetic algorithms perform better than simulated annealing in a complicated problem domain.

Tabu search method allows both non-improving moves and a deterministic choice of a solution from the neighborhood of the current solution. We declare solutions already visited as tabu, and move to the best non-tabu neighbor. This approach is not practicable, since it involves too much time and effort to store complete solutions in the tabu list and to compare other solutions with the elements of the tabu list. Thus nowadays, tabu search is rather a general framework than a concrete method. [Hurink, 2005]

10. Implication for Reverse Enterprise System

There is more complexity in RES operating as flexible systems with multiple products going through the forward and reverse logistics. Thus the product return process in RES can be seen as a flexible system where the proposed GA based tool can also be used. The RES network links initial collection points, centralization return centers, and manufacturing facilities etc.

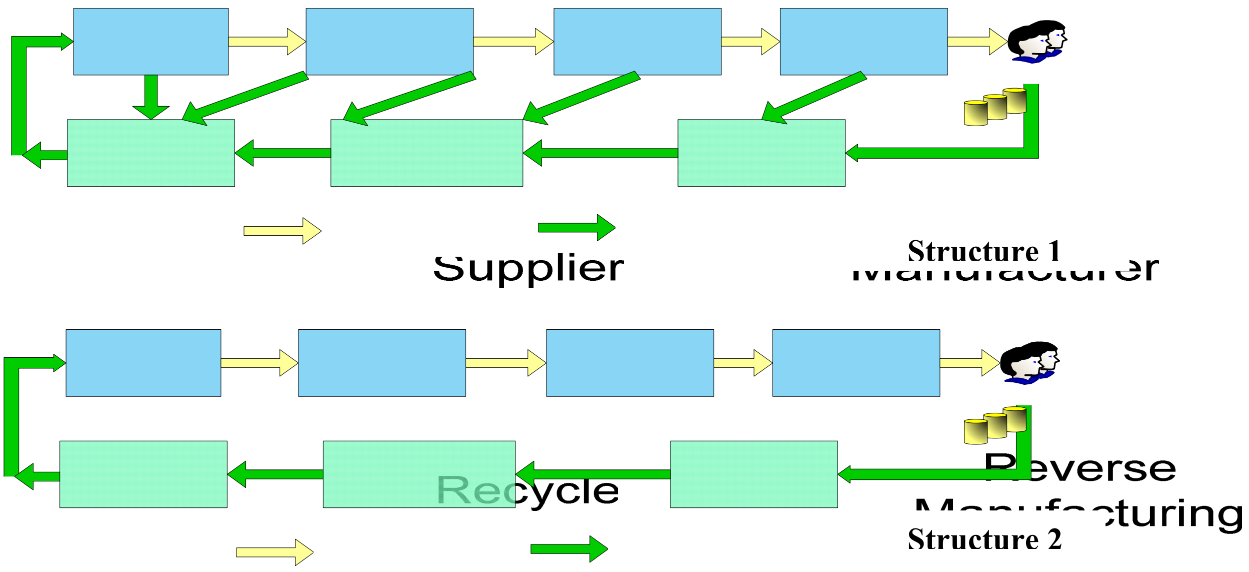


Figure 9.1. Representation of different structures for flexible flow line system for RES

A flexible mixed model flow line problem can be compared with various nodes in Reverse Enterprise System (RES) (Wadhwa & Madaan, 2004). Considering a simple RES system 'n' independent jobs acts as returned products model to the collection centre (Collector) required to be reprocessed on 'm' different re-processing facility as shown in Figure 9.1. Here the objective is to find the schedule for the returned products on the reverse manufacturing facilities (re-manufacturing, cannibalization, repair, reuse, resell, etc.) that minimizes the lead time. This reduction in lead time will reduce cost of time delays and has its significant effect on asset recovery in a Reverse Enterprise System (Gilvan, 2006). Suggested tool can be very effective in cases with a large variety of products following the same sequence on a large number of processing facilities. Therefore recognizing the inherent complexity of the product return process, proposed GA based tool can help to achieve improved performance of the reverse logistics network linking initial collection points, centralized return centers, and the manufacturing facilities in the system.

11. Conclusion

With the objective to minimize the total completion time (make-span or lead time), PFSP is very difficult to solve effectively due to the NP-hardness. In this paper we have proposed the use of GA for permutation flow-shop scheduling problem under the make-span minimization criteria.

Here architecture has been developed to depict the application of genetic algorithms to achieve improved production schedules in a flow-shop. Based on this architecture, a simulator integrated with genetic algorithm for solving the mixed model flexible flow-shop scheduling problems has been developed. The simulator has been incorporated with a user-friendly GUI, which acts as an interface between the end user and the simulator. Based on the input parameters entered by the end-user, the scheduler is capable of providing near optimal solutions, in the form of a sequence with minimum make-span, very quickly. Hence, this tool can be very effective in cases with a large variety of jobs following the same sequence on a large number of machines. Despite numerous merits, the proposed model and solution procedure point to a number of directions for future work. The model can be expanded to include the element of risk and uncertainty present in real manufacturing scenarios. Research is in progress to model more complex systems, for instance, the multilevel flexible product return system for requiring transport control decisions in RES. Our future research will involve the extension of this tool to include the element of risk and uncertainty involved in the reverse logistics network design problem. Various flexible system domains where GA based applications can play an important role, include, flexible manufacturing systems, multi-product supply chains, flexible supply chains, multi-product reverse logistics and reverse enterprise systems etc.

REFERENCES

1. BENJAAFAR, S. and RAMAKRISHNAN. R., **Modelling Measurement and Evaluation of Sequencing Flexibility in Manufacturing System,** Int. J. Prod. Res., Vol. 34, 1996, 1195–1220.
2. CONWAY, R.W., MAXWELL W.L. and MILLER L.W., **Theory of Scheduling**, Addison-Wesley, Reading, MA, 1967.
3. CARLSSON, B., **Flexibility and the Theory of the Firm.**, Int. J. Ind. Org., Vol. 7, 1989, 179–203.
4. CHAN, F.T.S., CHUNG S.H. and WADHWA S., **A Hybrid Genetic Algorithm for Production and Distribution**, Omega-International Journal of Management Science, Vol. 33, 2005, 345-355.
5. GAREY, M.R. and JOHNSON D.S., **Computers and Intractability: a Guide to the Theory of NP-completeness**, San Francisco, 1979.
6. GAREY, M.R., JOHNSON D.S. and SETHI R., **The Complexity of Flowshop and Jobshop Scheduling**, Mathematics of Operations Research, Vol. 1, 1976, 117-129.
7. GOLDBERG, D.E., **Genetic Algorithms in Search, Optimization and Machine Learning**, Addison-Wesley, Reading, MA, 1989.
8. GRAHAM, R.L., LAWLER E.L., LENSTRA J.K. and RINNOOY A.H.G., **Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey**, Annals of Discrete Mathematics, Vol. 5, 1979, 287-326.
9. GRUPE, F.H. and JOOSTE S., **Genetic Algorithms: A Business Perspective**, Information Management and Computer Security, Vol. 12, 2004, 289-298.
10. HOKEY, MIN, HYUN JEUNG KO & CHANG SEONG KO, **A Eenetic Algorithm Approach to Developing the Multi-echelon RL Network for Product Returns**, Omega, Vol. 34, No. 1, 2006, 56-69.
11. HURINK, J., **Introduction to Local Search**, http://www.home.math.utwente.nl/~hurinkjl/socra-text.pdf, University of Twente, 2005.
12. KLASSEN, R.D., ANGELL L.C., **An International Comparison of Environmental Management in Operations: the Impact of Manufacturing Flexibility in the U.S. and Germany**. Journal of Operations Management 1998; Vol. 16, No. 3-4, 1998, 177- 94.
13. OGBU, F.A. and SMITH D.K., **The Application of the SA Algorithm to the Solution of the *n/m/C*max flowshop problem**, Computers and Operations Research, Vol. 17, 1990, 243–53.
14. REEVES, C.R., **A Genetic Algorithm for Flowshop Sequencing**, Computers and Operations Research, Vol. 22, 1995, 5–13.
15. RUIZ, R., MAROTO C. and ALCARAZ J., **Two New Robust GA for Flowshop Scheduling Problem**, OMEGA, The International Journal of Management Science, Vol. 34, 2006, 461-476.
16. SADEGHEIH, A., **Scheduling Problem Using GA, Simulated Annealing and the Effects of Parameter Values on GA Performance**, Applied Mathematical Modeling, Vol. 30, 2006, 147-154.
17. SILVA, C.A., SOUSA J.M.C., RUNKLER T. and PALM R., **Soft Computing Optimization Methods Applied to Logistic Processes**, Internatl Jrnl of Approximate Reasoning, Vol. 40, No. 3, 2005, 280-301.
18. WADHWA, S. and AMIT CHOPRA, **A Genetic Algorithm Application: Dynamic Re-configuration in Agile Manufacturing Systems**, Studies in Informatics and Control, Vol. 9, 2000, 1-9.
19. WADHWA, S. and BROWNE, J., **Modelling FMS with Decision Petri Nets**, Int. J. Flex. Manuf. Syst., Vol. 1, 2005, 253–280.
20. WADHWA, S. and MADAAN, J., **Dynamic System Modeling for Closed Loop Supply Chain (CLSC) System**, INFORMS 2006 International Conference, June, Hong Kong, China.
21. WADHWA, S. and MADAAN, J., **Reverse Enterprise System: Evolving Perspective**, Third Global Conference on Flexible System Management, March, New Delhi, 2004.

22. WADHWA, S. and RAO, K.S., **Flexibility: An Emerging Meta-competence for Managing High Technology**. Int. J. Technology Management, Vol. 19, 2000, 820–845.

23. WADHWA, S., RAO K.S, and CHAN F. T. S., **Flexibility-enabled Lead-time Reduction in Flexible Systems**, International Journal of Production Research, Vol. 43, No. 15, 2005, 3131–3162.

24. WADHWA, S., RAO K.S, and CHAN F. T. S., **Comparative Influence of Three Flexibility Types on Manufacturing Lead-time Performance**, Vol. 29, No. 9-10, 2006, 1002–1007.

25. WANG, L., ZHANG L. and ZHENG D.Z., **An Effective Hybrid Genetic Algorithm for Flow Shop Scheduling with Limited Buffers**, Computers & Operations Research, Vol. 33, 2006, 2960 – 2971.

26. WANG, L. and ZHENG D.Z., **An Effective Hybrid Heuristic for Flow Shop Scheduling**, International Journal of Advanced Manufacture Technology, Vol. 21, 2003, 38–44.

27. WHITLEY, D., **A Genetic Algorithm Tutorial**, Colorado State University, Fort Collins, 2005, http://samizdat.mines.edu/ga_tutorial

28. WU, L., **A Comparison of Nature Inspired Intelligent Optimization Methods in Aerial Spray Deposition Management**, 2002, http://www.cs.uga.edu/~potter/dendrite/wu_lei_thesis.ppt, University of Georgia.