# Characterising Genetic Algorithm Approaches to Job Shop Scheduling

**Bhuvan Sharma**
Computing, Engineering and Mathematical Sciences
University of West England, Bristol
Bhuvan.sharma@uwe.ac.uk

**Xin Yao**
School of Computer Science
University of Birmingham
X.yao@cs.bham.ac.uk

## Abstract

In most production environments, there is a requirement for effective use of the available resources. Job shop-scheduling problem (JSSP) because of its key impact on revenues is at the crux of Production planning, of late also known as Enterprise Resource Planning (ERP). Being an NP complete and highly constrained problem, exact methods take exponential time and heuristic methods give suboptimal solutions. Genetic Algorithms (GA) are the most sought after techniques to get global optimal solutions, and this paper reviews various GA approaches to solve JSSP.

The parameters, representation schemes, and operators used in a GA strongly affect the quality of search. The paper starts with analysis on the use of heuristics for JSSP, and it is pointed out why evolutionary approaches are better than heuristics. This is followed by a detailed discussion on various representation schemes for a feasible schedule. These schemes are reviewed in terms of their ease of implementation in GA, choice of crossover and mutation operators, and their commensurate advantages and disadvantages. In the end some schemes are analyzed as to their suitability in improving the performance of GA's. The motivation behind such a paper is to lay out a critical discussion on the GA methodologies for JSSP, based on representation schemes, in order to better serve the needs of manufacturing community trying to apply GA on their problems.

## 1 Introduction

A general job shop scheduling problem (*JSSP*) can be defined as n job, m machine problem, where the aim is to optimally allocate series of operations for each job across available machines, respecting temporal and resource constraints.

Informally the problem can be described as follows: There is a set of jobs (*Ji*) and a set of machines (*Mi*). Each job has a technological sequence of operations through one or more of the machines, as in Figure 1. Each operation is of fixed time interval ($P_{ir}$). The processing of job $J_i$ on machine $M_r$ is called the operation $O_{jr}$. Operation $O_{jr}$ requires the exclusive use of machine $M_r$, for time duration $P_{jr}$, its processing time. The time required to complete all the jobs is called the make-span. The processing order of operations for a job, as well as the processing time of each operation is known.

*Schedule* is allocation of the operations to specific time intervals on machines. Often the objective is to find a schedule of minimum length. There are other less important objectives associated with JSSP, some of which are: minimization of delay, maximization of work in progress, minimization of idle time. The sequence in the present paper would be an introduction to JSSP's and Genetic Algorithms, followed by an extensive review and analysis of various GA approaches.

### 1.1 Complexity of JSSP

JSSP is worst among the NP hard problems and among the worst combinatorial problems as in Garey, et al [1]. An indication for this is that one 10X10 problem formulated by Muth, et al. [2] remained unsolved for 20 years.

Any JSSP is more than often associated with constraints. There is a pre-specified sequence of machining operations *(Ojr)* through which each job has to be processed, and a fixed processing time *(Pjr)* for each such operation. Keeping these constraints in mind the objective is to determine when an operation is to be allocated to the machine in order to minimize the make-span.

| | | | |
|---|---|---|---|
| J1 | $M_1, P_{11}$ | $M_2, P_{12}$ | $M_3, P_{13}$ |
| J2 | $M_1, P_{21}$ | $M_3\ P_{23}$ | $M_2, P_{22}$ |
| J3 | $M_3, P_{33}$ | $M_2, P_{32}$ | $M_1, P_{31}$ |

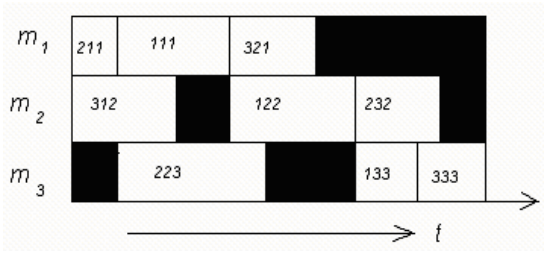
Figure 1: 3 X 3 JSSP

A scheduling problem can increase in complexity and more and more real world job shop scheduling problems have further constraints and flexibilities as described in Todd et al. [3]. For example there could be more than one machine where a particular operation of a job could be performed, and/or there

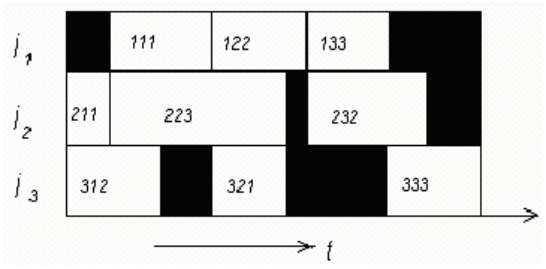
could be interdependencies between jobs, and/or a machine can perform more than one type of operation. Such problems require more focused approach, however the context of current work focuses on a broader paradigm of Genetic Algorithm approaches to JSSP. The techniques described are generic and can be extended to complex problems as well.

### 1.2 Representation of schedule

Feasible schedules can be illustrated by means of Gantt charts: machine Gantt chart (times for which job operations visit a machine) and job Gantt chart (times for which machines are allotted to job operations) as shown in Figure 2.




(a) Machine Gantt chart




(b) Job Gantt chart

*Figure 2: Representation of solution in JSSP*

In figure 2, the index used for the operation is *jom* where *j* represents the job, *o* indicates operation of job *j*, and *m* represents machine to perform this operation. In principle there are infinite feasible schedules for a job shop problem, because idle times can be inserted between two operations. But the aim is always to shift the operations to the left as compact as possible. A shift in a schedule is called a local left shift if some operations can be started earlier in time without altering the operation sequence. A shift is called global left shift if some operations can be started earlier in time without delaying any other operation sequence even though the shift has changed the operation sequence.

### 1.3 Genetic Algorithm

Genetic Algorithms, proposed by Holland [4] are stochastic, guided search strategy. They work on a population of points, and follow the Darwinian principle of survival of fittest in nature. A population of solutions is evolved to a next set of solutions through genetic operators, and mechanism of selection just like the process of evolution in nature. The idea is to get a new population with better desired characteristics then the previous. The process is repeated for the newly obtained set of solutions, until a desired level of comfort in terms of termination criterion is achieved.

Genetic Algorithms are especially suited for JSSP, not only because the problem is multi-objective, and constrained, more so because of the deceptive nature of fitness landscape. It is these aspects of the problem which limit the application of other techniques like tabu search as in [5,6,7,8,9,10,54,55] and simulated annealing as in Laarhoven et al. [11], and/or branch and bound as in Baker et al. [12] and Brah et al. [13]. Many of these approaches become ineffective for large timetabling and scheduling problems.

GAs have frequently been applied for combinatorial optimisation problems such as bin packing, Falkenauer, et al.[4], machine sequencing, Reeves [5], vehicle routing, Blanton, et al. [6] and traveling salesman problems. JSSP is another class of combinatorial problem; more complicated than the previous listed problems because often it involves much more constraints.

## 2 Genetic Algorithm vs. Heuristics

Heuristic techniques are often motivated by the question 'what is the best way to generate a good solution' while genetic algorithms target the specific question 'what is the best solution to the problem'. To this effect, heuristic is generally a rule giving priority to one operation over other. A comprehensive survey of heuristic rules is found in Panwalker, et al. [14], where 113 rules are presented, and in [15,16,17]. These rule based techniques are limited because it was argued in Savell, et al. [18] that as the production environment complexity increases the rules tend to become less effective. Besides, choosing only one of the rules throughout the production schedule would result in less robust behavior as shown by Fisher, et al. [19], where it was shown that choosing from multiple rules was better than sticking to a single rule, also stated originally by Jeremiah, et al. [20]. Even after selecting multiple rules no global optimal solution was found. The various heuristic approaches can be found in [21,22,23,49,50,51,52,53].

## 3 Encoding JSSP for GA

One of the critical requirements in GA is the representation of the attributes (genes) that make up an individual solution (chromosome) of the population.

Broadly speaking various representation approaches can be classified into two main categories: direct and indirect representation.

In indirect representation the chromosome encodes a sequence of preferences, and a schedule builder is often required to decode the chromosome to a schedule. The idea behind indirect representation is to evolve the sequence of decision preferences by means of genetic operators. These decision preferences can be heuristic rules or simple ordering of jobs in a machine. In, direct representation the chromosome directly encodes a schedule making schedule apparent and thereby eliminating the need for a complex schedule builder. At the same time, applying simple straightforward genetic operators on direct representation string often give infeasible schedules hence such a technique requires domain specific recombination operators, or some kind of repair mechanisms.

Examples of indirect representation are traditional binary representation, priority rule based representation and preference list based representation. Job based representation, and operation based representation are examples of direct representation. All these representations are discussed in the perspective of ease of use of operators and possible implications.

### 3.1 Binary Encoding of Schedule

The most obvious indirect representation approach is the conventional binary coding. However, conventional mutation and crossover can be applied here without any difficulty. JSSP was tackled using binary representation in Nakano [24]. The approach focuses on a job pair [j,k] and the machining sequence of operations for these jobs. The operation sequence within j and k could be represented as [*Oj1, Oj2…..OjM*] and [*Ok1, Ok2, …..OkM*] where numeral *1, 2, 3 Mi,* etc indicate operation number. It is not necessary that Oj1 and Ok1 be carried out on the same machine.

Then for all the operations of these job pair that are executed on the same machine, one gets a bit vector of the form

*[prior(Oj1, Ok*), prior(Oj2,Ok*)….prior(OjM, Ok*)]*

where *prior(Oj1, Ok*)* = 1 or 0, depending whether operation *Oj1* is carried prior or later to operation Ok* on the machine on which operation 1 of job j is carried out. Figure 2 shows how a symbolic solution is arrived at, given the machining sequence and binary representation for a 3X3 problem.

| | | | | | |
|---|---|---|---|---|---|
| Job 1 | M1 | M3 | M2 | Job1 & 2 | 1 1 0 |
| Job 2 | M2 | M3 | M1 | Job1 & 3 | 1 1 0 |
| Job 3 | M2 | M1 | M3 | Job2 & 3 | 0 1 0 |

a) Machining Sequence b) Binary Representation

| | | | |
|---|---|---|---|
| M/c 1 | J1 | J3 | J2 |
| M/c 2 | J3 | J2 | J1 |
| M/c 3 | J1 | J2 | J3 |

c) Symbolic Representation

Figure 3: *The conventional Binary Representation Approach for JSSP*

The resulting representation is a matrix of rows equal to number of job pairs and columns equal to number of operations per job. For N job M machine the matrix would contain M*N(N-1)/2 binary bits. The number comes out to be huge since for as simple 6X6 problem the number of bits would be 90.

Demerit of such a scheme primarily is the redundancy in representation causing overhead in terms of penalty functions and/or repair strategies. Also this approach is less suited for scheduling problems with too many constraints as suggested in Holland [4], and Goldberg [25].

The obvious merit of such a scheme is the ease of using standard crossover and mutation operators.

### 3.2 Preference List Based Representation or Job Sequence Matrix

The method also sometimes refereed to as Partitioned Permutation Encoding, uses a chromosome consisting of *m* sub-chromosomes where *m* is the number of machines. Falkenauer, et al [11], used it for job shop scheduling problem with release times and due dates. This kind of representation was first presented by Davis [26], for a kind of scheduling problem and was also used by Croce, et al. [27], for classical job shop problems.

A schedule is represented by the set of permutations of jobs on each machine, or in other words a *m* partitioned permutations of operation numbers, thereby getting a *job sequence matrix.* A possible representation for a 3X3 problem could be:

| M1 | M2 | M3 |
|---|---|---|
| 132 | 321 | 213 |

The integers denote the job number. The first sub-chromosome (132) is the preference list for machine M1. In the event that a job cannot be send to a machine, the next from the list is chosen and so on. This approach requires a robust schedule builder, but eliminates the problems of redundancy and repair approach which were present in binary encoding scheme above.

Also this scheme of representation needs special crossover techniques. Some like Subsequence

Exchange Crossover (SXX) developed by Kobayashi, et al. [28], linear order crossover by Falkenauer, et al. [29] and Job order based crossover (JOX) as in Yamamura et al. [30] are best suited for Partitioned permutation encoding.

Any crossover in this scheme should take care that there is no repetition of integers in each of the sub-chromosome. SXX is a natural extension of sub-tour exchange crossover for traveling salesman problem used by Kobayashi, et al. [31]. The name subsequence comes because a sub-sequence of jobs is selected from each sub-chromosome of the two parent chromosomes, making sure that the subsequence string of jobs should be *exchangeable.* This means that the jobs in the subsequence selected should be same, though they can be in different permutated form. The working of SXX is detailed below:

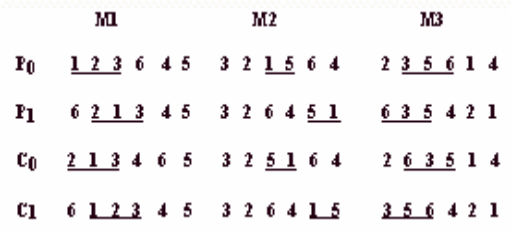



Figure 4: Subsequence Exchange Crossover *(from Kobayashi, et al. [28])*

The two offspring are obtained by exchanging such sub-sequences between two parents for each machine. One problem with this crossover is the inherent overload in selecting a substring from each sub-sequence which satisfies the requirement of being present in both parents in same or different permuted form. If no such string is found, then the crossover would effectively exchange the whole sequence, thereby in effect causing no change in the fitness. However the potential advantage of SXX is that it helps moving the solutions stuck in local optima. This is so because two solutions which are sub-optimal, must have a good sub-sequence and exchanging these will help moving away from local optima.

Linear order crossover is an extension of the order crossover used in TSP, in the sense that the information about high and low priority operations is not lost during the crossover. The working is illustrated below:

The approach to such a crossover is by choosing two points on a parent, taking the genes out between these two points and inserting them at exactly the same place in other parent, keeping in mind that such exchange of genetic material does not cause repetition in other parent. To this effect, the genes in parent two are reordered, by first removing the genes that are taken from the parent one and empty spaces so formed rearranged to accommodate the genes from parent one. Such a technique preserves the internal order of exchanged genes, and is an extension to SXX, in which often no crossover is possible between two sub chromosomes of parents if there is no substring of same numbers. Linear order crossover guarantees a crossover, but during the later stages when the population would converge, this crossover will not help in exploration. Also this method preserves the priority order of jobs, in the two parents.

Job based order crossover (JOX) as in figure 4 was suggested by Yamamura et al. [30] for Preference List Based Representation scheme and is an extension to uniform crossover as in binary coded GA or EP. Random jobs are selected and their locus in the sequence is fixed in both the parent. Child 1 and child 2 get these jobs from respective parent with their locus unaltered. Rest of the jobs in child 1 are transferred from parent 2 and those in child 2 are transferred from parent 1 in their respective order of occurrence. As in figure 4, jobs 2 and 4 are selected randomly and their position unchanged between parent and child, while rest of the string is populated from jobs in other parent.

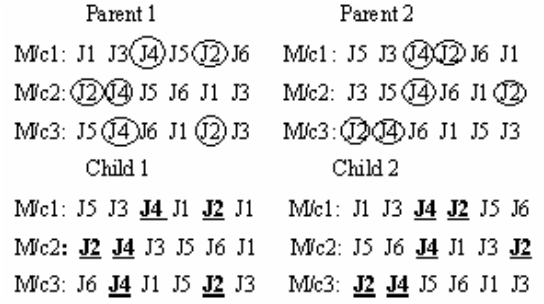



Figure 5: Job order based crossover (JOX) (*from Yamamura et al. [30]*)

This scheme would give infeasible solutions and Yamamura [RR] used GT algorithm to yield legal schedules. However the crossover preserves the order of each job on all machines between parents and their children.

### 3.3 Priority Rule Based GA

Priority rules are type of heuristic rules, which tend to guide the scheduling process. The operation is selected on the basis of these rules. In this representation, a chromosome is a string of genes where each gene represents one of the heuristic priority rules.

Let there be k operations say *O1*, *O2, ...Ok*, and depending on the partially built schedule, each operation has the earliest starting time (EST) given by *Oit.* Besides that each operation has a processing time given by *Oip*, and a flag *Oig,* which takes value

either 0 or 1. If *Oig* is one than the operation fits in the gap on the particular machine, if it is 0 then it doesn't. Now the conflicting set of operations are the ones whose *Oig* is 1, and the task of selecting one out of these is carried out by one of the priority rules. Some of these rules could be:

- Shortest Processing Time (SPT): An operation with minimum *Oip* gets preference over other conflicting operations. The motive is not to let operations wait for long.
- Longest Processing Time (LPT): An operation with maximum *Oip* gets preference over other conflicting operations. It is motivated by the desire of not leaving operations with large processing times till the end.
- Critical ratio (CR): It is an index number computed by dividing the time remaining until due date by the work days remaining. Since it takes care of urgency in certain operations by prioritizing jobs that must be finished to keep shipping the completed jobs, it has wide applications in industrial scheduling.
- Earliest Due Date (EDD): The job with the earliest due date is assigned to the schedule with highest priority.
- Smallest Remaining operations (SRO): The operation with smallest number of subsequent operations remaining.
- Largest Remaining operations (LRO): The operation with largest number of subsequent operations remaining.

Some of the other not so common rules are given in Sauer [32].

A chromosome typical of such a representation scheme would be a string of length *mn-1*, (*m x n* problem) and each gene would represent one of the priority rules as illustrated below.

[ SPT, LPT, EDD, CR, LRO, ….. ]

The length of string is *mn-1* because when just one job is left there wouldn't be any conflicting operations and no need of priority rule.

The advantage of this scheme is the ease of use of simple one point or two point crossover. Even crossover operators PPX, SXX and LOX described above can be used. This representation scheme is a clever approach to bring in the knowledge of the scheduling problem at hand into GA.

The disadvantage however is that the approach is not in the true spirit of GA, because priority rules might cause premature convergence. The situation gets aggravated if first few genes converge quickly, making the search for a globally optimal solution difficult.

The approach was used by Dorndorf, et al [33], and Pesch [34], and the schedule builder used a kind of algorithm first proposed by Giffler, et al. [22]. The steps in Giffler Thompson algorithm are detailed below. *ES(O)* and *EC(O)* denote the earliest starting time and the earliest completion time respectively for an operation O:

1. Let *D* be a set of all the earliest operations in a technological sequence not yet scheduled and *Ojr* be an operation with the minimum *EC* in *D: Ojr* arg min{O ∈ D | *EC(O)*}.
2. Assume i-1 operations have been scheduled on *Mr*. A *conflict set C[Mr; i]* is defined as: *C[Mr; i]* = {*Okr* ∈ *D* | *Okr* on *Mr* , *ES (Okr)* < *EC (Ojr)* }
3. Select an operation *O* ∈ *C[Mr , i]*.
4. Schedule *O* as the i-th operation on *Mr* with its completion time equal to *EC(O)*.

### 3.4 Job Based Representation

The method, as in Holsapple, et al. [35], uses a list of n jobs: the schedule from which is constructed according to the sequence of jobs. The chromosome is a set of permutation of jobs, defining the scheduling priority of each job with respect to others. Note that scheduling priority is different from processing priority.

For example a typical chromosome for a 6 job problem could be:

[ 4 3 5 6 1 2]

All operations for job 4 are scheduled first followed by all of job 3 and so on, in the possible space of time.

For a 3X3 problem, given in figure 1 and the job based chromosome representation as

[ 2 1 3 ]

the schedule generated would be as shown in Figure 6.

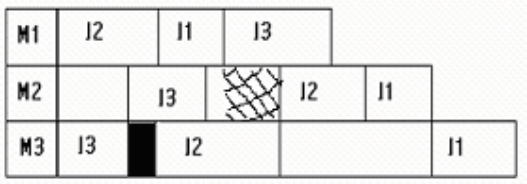



*Figure 6: Schedule generated from a Job Based representation of [213] for 3X3 problem of figure* 1.

The job 2 is sequenced on the machines first, followed by job 1. Note that jobs are scheduled but not necessarily processed in that order. Also note that the second operation for job 1 cannot be scheduled on machine 2 immediately after first operation on machine 1 is finished. This is because the available time gap (shaded square in figure 6) between finish of operation 1 and start time of already scheduled operation of job 2 on machine 2 is less than the processing time of this 2nd operation of job1.

After the final operation of job 1 is scheduled, job 3 is scheduled. Scheduling of operation 1 of job 3 on m/c 3 before the already scheduled operation 2 of job 2 is feasible, since available time on m/c 3 before the

2nd operation of job 2 is in excess of the processing time of operation to be scheduled. . The excess time is denoted in Figure 6 by dark portion.
The clear disadvantage of such a scheme is that it is highly constrained because of the fixation in job scheduling priority. This may lead to suboptimal solutions. The advantage is in the simple schedule builder required, because any representation will yield a feasible schedule.

### 3.5 Operation Based Representation

Bierwirth [36], used an un-partitioned operation based representation where each job integer is repeated m times, where m is the number of machines. The $k^{th}$ occurrence of a job number would refer to the $k^{th}$ operation of that job. The sequence of operations thus represents the priority in which they are to be scheduled.

For a n x m JSSP the length of the chromosome would be n x m. A typical chromosome for such a representation and for a 3x3 problem could be:

[1 2 2 3 1 3 2 1 3]

The decoding of such a chromosome into a feasible schedule is done as shown below in Figure 7.

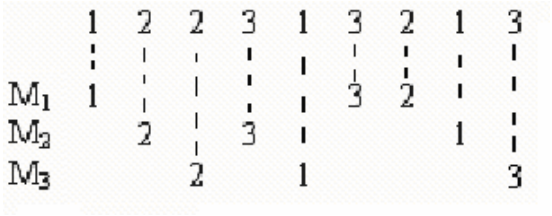



*Figure 7: Genotype for un-partitioned permutation approach for a 3X3 problem*

The decoded schedule is obtained by the schedule builder, and all the schedules thus produced are valid. In the example shown in Figure 7 each of job is repeated 3 times since there are 3 operations per job. Respecting the constraints as in Figure 2 and taking job 2 for example, the first occurrence of 2 would mean allocating it to machine 2, second occurrence would mean allocating it to machine 3 and then to machine 1. This way the processing order of jobs on each machine is deduced. A similar scheme of representation is also found in Fang, et al. [37].

The advantage of such a scheme is that it requires a very simple schedule builder because all the schedules generated are legal. The working of schedule builder in this case is straight forward, since only a check of available time slot on the machine is to be considered while allocating an operation to a particular machine. Another aspect of such a representation is that the numbers of possibilities explored by the genotype are:

$$\frac{M1 + M2 + M3 + \ldots\ldots Mn}{!M1 * !M2 * !M3 * \ldots\ldots !Mn}$$

where M1, M2, etc denote the number of machines the jobs 1, 2 and 3 would visit in the entire schedule. This space of genotype covers all of the active schedules possible, though it brings in some redundancy because actual number of possible schedules are only *!M1 X !M2 X !M3*.

A simple one point or two point crossover does not apply in this case, because the resultant schedule is not always representative. Each job is allowed to repeat exactly the number of times equal to the machines on which it will be processed. This constraint is difficult to maintain in one point and two point simple crossovers.
Two crossover techniques are implemented that preserve the constraints. The Generalized order crossover (GOX) an extension of Order crossover for simple permutation schemes by Oliver, et al. [38] is generalized to make it suitable for this scheme.
In GOX the donor parent contributes a substring of length normally in the range of one third to half of the length of the chromosome string. Choosing such a length of substring ensures that the offspring will inherit almost the same amount of information from both the parents. As shown below, the genes in the selected substring are also given an index number which states the occurrence number of that particular operation of the job.
*Chromosome:* *A B B A C A B C B C*
*Index number of genes in substring* *2 1 3 3*
To determine a position where to implant the string, a method known from order crossover used in TSP is used. The genes in the receiver chromosome that are representative of the substring in terms of index number are marked and the parent gets a cut at the gene which is equal to the first gene (and its index) in the donor crossover string. All the marked genes are then removed. If the receiver parent is as shown below with the marked genes in bold,
B A B **B C *A*** C C B **A**
the offspring will get the substring A C A B after the ***A*** marked with italics and the resultant offspring will be:
B A B A C A B C C B
Another crossover known as Precedence preservation crossover (PPX) proposed by Bierwirth, et al. [39] also respects the absolute order of genes in parental chromosomes. As in Figure 8 a template binary string of 0's and 1s is used to define the order in which genes are drawn from $P_0$ and $P_1$. A 0 would mean draw a gene from parent $P_0$, while occurrence of 1 would mean draw a gene from parent $P_1$. As a gene is drawn from one parent it is appended to the developing offspring, and the corresponding gene from other parent is deleted. This step is repeated

until both parent chromosomes are exhausted, and consequently the offspring contains all the genes.

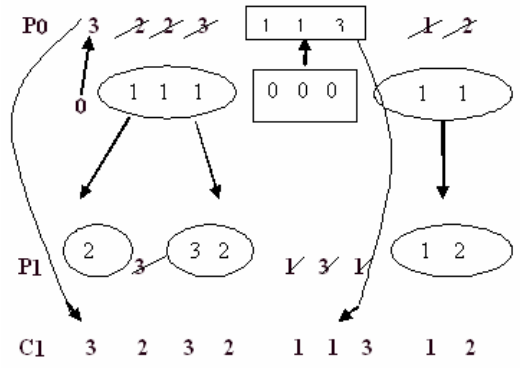



*Figure 8: Precedence preservation Crossover (PPX) (from Bierwirth, et al.[39])*

This scheme leads to efficient genetic local search as in Bierwirth, [36]**.** The results of the experiments run on benchmark problems showed that the average solution was 0.7 to 7.2 % of the best known solution, while the best solution using this approach was 0-4.5% of the best solution known so far for these problems.

The GOX and PPX very effectively preserve the characteristics of both the parents. GOX preserves the repetition structure of certain alleles of both parents. The PPX respects the absolute order of genes in the parents. This is reflected by the hamming distance, of the offspring from both parents, which is the same as shown in Bierwirth, et al. [39]. The operators also take care of yielding valid schedules.

One drawback of above scheme of representation was pointed in Song, et al. [40], stating that the last *m* bits of chromosome were redundant, and consequently a shorter chromosome was used in search. The motivation behind the work was that last few genes least affect the solution quality, and instead drag the evolution process, so a heuristic approach was used to select the last *m* bits of string.

## 4 Performance Improvement in Search

Several works indicate that GAs are not well suited for fine tuning structures that are very close to optimal solutions as in Dorndorf, et al. [33], and Bierwirth [36] as crossover operators generally lose their efficiency in order to generate feasible schedules. Certain techniques are applied either in conjunction with genetic algorithms or sometimes targeted at the operator level of genetic algorithms.

To this effect, Genetic Local Search (GLS) (or population based local search or memetic search) as in Grefenstette [41], Moscato [42], and Ulder *et al*. [43] is applied which incorporates local search neighbourhoods into the GA strategy.

In memetic approach a solution conceived from the GA operators is used as the starting point for subsequent improvement through neighbourhood search by small perturbations, and process continued either for specified number of perturbations or until a locally optimal solution is obtained. The improved solution becomes the member of next set of population and then operated on in the next generation by the traditional genetic recombination operators.

The superiority of GLS over GAs is highlighted in Della Croce *et al*. [44]. Various neighbourhood structures for the genetic algorithm approaches to JSSP can be found in Della Croce *et al*. [45]. One of the most well known GLS works is by Dorndorf and Pesch [33] where an approach called P-GA is proposed to solve JSSP. The method guides a probabilistic combination of 12 priority despatch rules. A similar chromosome evolution framework is applied by Pesch [46] who applies a GLS strategy to control subproblem selections in his decomposition approach. The subproblems are solved by a constraint propagation approach which finds good solutions by fixing arc directions.

The best GLS methods appear to be those integrated with other techniques, for example Mattfeld [47] with spatial populations and attitude inheritance and Yamada and Nakano (1996b, c) with critical neighbourhoods and various forms of acceptance probabilities.

Another novel approach called GVOT (Gene Variance Based Operator Targeting) was proposed in Fang, et al. [48]. According to them, fast stabilization of early parts of genome leads to premature convergence. GVOT works by measuring the diversity of genes at each position of the genotype. In their experiments the diversity in the genes is calculated by measuring their statistical variance after every ten generations, and based on these variances the actual point of crossover or mutation is chosen. The higher the variation in a chunk, higher will be the probability for crossover and vice versa for mutation. The performance against the benchmark 10X10 and 20X5 was found to be better than the simple binary approach without GVOT as in Nakano et al. [24].

## 5 Conclusion

A critical review of various Genetic Algorithm approaches to Job shop scheduling problem was presented. The case for GA's in preference to heuristic was argued given the fact that no heuristics with a guaranteed performance has been developed so far. Furthermore, given that GA are inspired by idea of globally optimal solution as opposed to heuristics should make them a preferred choice.

Various GA techniques were critically reviewed and comparisons drawn in terms of their representation and commensurate ease of use of schedule builder and crossover operators. It can be argued that no single technique exists without any drawbacks. In less complex representations (job based, priority rule based) the simple representation was offset by the resultant convergence to sub optimal solutions. More complex representations (binary, preference list based) though gave redundant solutions and required robust schedule builder and crossover operators, yet these things were offset by better exploratory search process. It can finally be argued that various factors in GA can affect the quality of search and thereby the quality of algorithm and final solutions. Some of the parameters like type of crossover is representation specific, and thereby choosing representation is a crucial factor, because a good representation scheme can help avoid infeasible schedules as well as avoid premature convergence.

Towards the end a few lateral approaches, in the form of hybrid algorithms (GLS) and other methodologies (GVOT) were discussed that would enhance the performance of Genetic search process within the GA.

## References


[1] M. Garey, D. Johnson and R. Sethi, (1976). The complexity of flow shop and job shop scheduling, in *Maths Ops Res*. 1, pp 117-129.

[2] J.F Muth, and G.L Thompson (1963). Industrial Scheduling. *Prentice-Hall, Englewood Cliffs, N.J*

[3] D.S Todd, P. Sen, (1998). Tackling Complex Job Shop Problems Using Operation Based Scheduling, in *24th International Conference on Computers and Industrial Engineering.*

[4] J.H. Holland (1975). Adaptation in Natural and Artificial System. *University Of Michigan Press.*

[5] M. Gopalakrishnan, S. Mohan, Z. He.(2001). A tabu search heuristic for preventive maintenance scheduling, in *Journal of Computers & Industrial Engineering*

[6] V. A. Armeta, C. R. Scrich(2001). Tabu search for minimizing total tardiness in a job shop., in *International Journal of Production Economics*.

[7] J. Blazewicz, W. Domschke, and E. Pesch (1996). The job shop scheduling problem: Conventional and new solution techniques, in *European Journal of Operational Research*, 93:1–33.

[8] A.S. Jain and S. Meeran (1999). Deterministic job-shop scheduling: Past, present and future, in *European Journal of Operational Research*, 113:390–434,.

[9] Eric D. Taillard (1994). Parallel taboo search techniques for the job shop scheduling problem, in *ORSA Journal on Computing*, 6(2):108–117.

[10] R.J.M. Vaessens, E.H.L. Aarts, and J.K. Lenstra (1996). Job shop scheduling by local search, in *INFORMS Journal on Computing*, 8(3):302–317.

[11] P. J. M. Laarhoven, E. H. L. Aarts, and J. K. Lenstra. (1992). Job shop scheduling by simulated annealing, in *Operations Research, 40(1):113-125.*

[12] J. R. Baker and G. B. McMahon (1985). Scheduling the general job-shop. *Management Science, 31(5):594--598, 1985.*

[13] S.A. Brah, J.L. Hunsucker (1991). Branch and bound algorithm for the flow shop with multiple processors, in *European Journal of Operations Research 51 88–99.*

[14] Panwalkar, S. S. and Iskander, W. (1977) A Survey of Scheduling Rules, in *Operations Research*, Jan-Feb, 25(1), 45-61.

[15] Blackstone, J. H. Jr, Phillips, D. T. and Hogg, G. L. (1982) A State of the Art Survey of Dispatching Rules for Manufacturing Job-Shop Operations, in *International Journal Of Production Research*, Jan-Feb, vol 20, 27-45.

[16] Bhaskaran, K. and Pinedo, M. (1991) Dispatching, In Salvendy, G. (ed.) *Handbook of Industrial Engineering*, *John Wiley and Sons, New York, chapter 83.*

[17] Haupt, R. (1989) A Survey of Priority Rule Based Scheduling, in *Opn. Res. Spektrum*, vol 11, 3-16.

[18] Savel, D. V., Perez, R. A., & Koh, S. W. (1989). Scheduling semi-conductor wafer production: an expert system implementation. *IEEE expert*, 9-15.

[19] Fisher, H., & Thompson, G. L. (1963). Probabilistic learning combinations of local job-shop scheduling rules. Muth J., & Thompson, G. (eds), *Industrial Scheduling, Englewood Cliffs, N.J., Prentice Hall*, 225--251.

[20] B. Jeremiah, A. Lalchandani and L. Schrage (1964). Heuristic rules towards Optimal Scheduling, in *Research Report, Department of Industrial Engineering, Cornell University.*

[21] M. Asano, H. Ohta. (2002). A heuristic for job shop scheduling to minimize total weighted tardiness, in *Journal of Computers & Industrial Engineering*, vol 42, Issue 2-4.

[22] Farzad Mahmoodi, Kevin J. Dooley, Patrick J. Starr (1990). An evaluation of order releasing and due date assignment heuristics in a cellular manufacturing system, in *Journal of Operations Management,* vol 9.

[23] M. Gopalakrishnan, S. Mohan, Z. He.(2001). A tabu search heuristic for preventive maintenance scheduling, in *Journal of Computers & Industrial Engineering,* vol 40, Issue 1-2.

[24] R Nakano, and T Yamada (1991). Conventional Genetic Algorithms for Job Shop Scheduling Problems*, in Proceedings of the 4th International Conference on Genetic Algorithms.*

[25] D.E. Goldberg (1986). Genetic Algorithm in search, optimization, and Machine Learning. *Addison-Wesley, Reading, Mass.*

[26] L. Davis (1985). Job shop Scheduling with genetic algorithms in *Proceedings of the International Conference on Genetic Algorithms and their applications*: pages 136-140.

[27] F. Croce, R Tadei and G. Volta (1995). A genetic algorithm for job shop problem, in *Computers and operations Research*, vol 22, pages 15-24.

[28] M. Kobayashi, T. Ono and M. Yamamura (1995). An efficient genetic algorithm for job shop scheduling problems, in *6th International Conference on Genetic Algorithms:* pages 506-511.

[29] Falkenauer, E. and S. Bouffoix, A genetic algorithm for job shop, in *Proc. 1991 IEEE Inter. Conf. on Robotics and Automation,* pp. 824- 829, 1991.

[30] M. Yamura, S. Kobayashi, I. Ono. Algorithm for Job-shop Scheduling Problems Using Job-based Order Crossover, in *Proc. of ICEC'96*, pp.547-552 (1996)

[31] M. Kobayashi, T. Ono and S. Kobayashi (1992). Character preserving genetic algorithms for traveling salesman problem, in *Journal of Japanese Society for Artificial Intelligence*: pages 1049-1059.

[32] Sauer, J. (1999). Knowledge-based systems, techniques and applications in Scheduling, in Leondes, T.L. (Ed.), *San Diego, Academic Press.*

[33] U. Dorndorf and E. Pesch (1995). Evolution based learning in a job shop scheduling environment, in *Computer Operations Research:* pages 25-40.

[34] E. Pesch (1994). Learning in Automated manufacturing: a local search approach, in *Physica-Verlag, Heidelberg, Germany.*

[35] C. Holsapple, V. Jacob, R. Pakath, and J Zaver (1993).. A genetic based hybrid scheduler forgenerating static schedules in flexible manufacturing contexts, in *IEEE transactions on System, Man, Cybernetics,* vol. 23, page 953-971.

[36] C. Bierwirth (1995). A generalized permutation approach to job shop scheduling with genetic algorithms, *in OR Spektrum*: page 87-92.

[37] H. Fang, P. Ross, and D. Corne (1993). A promising Genetic Algorithm approach to job shop scheduling, rescheduling and open shop scheduling problems, in *Proceedings of the* 5th *International Conference on Genetic Algorithms:* pages 375-382.

[38] I. Oliver, G. Smith, and J Holland (1987). A study of permutation crossover operator on the traveling salesman problem, in *Proceedings of the 2nd International Conference on genetic algorithms, Lawrence Erllbaum Associate:* pages 224-230.

[39] C. Bierwirth, D. Mattfeld, and H. Kopfer (1996). On permutation representations for scheduling problems, in *4th PPSN,* pages 310-318.

[40] Y. Song, JG. Hughes, N. Azarmi and C. Voudouris (*1999)*. A Genetic Algorithm with an incomplete representation for the Job Shop Scheduling Problems, in *The 18th Workshop of the UK Planning and Scheduling Special Interest Group, dec 15-16.*

[41] Grefenstette, J. J. (1987) Incorporating Problem Specific Knowledge into Genetic Algorithms, in Davis, L. (ed) *Genetic Algorithms and Simulated Annealing*, Pitman, pp. 42-60.

[42] Moscato, P. (1989) On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms, in *C3P Report 826, Caltech Concurrent Computation Program, Caltech, California, USA.*

[43]Ulder, N. L. J., Aarts, E. H. L., Bandelt, H.-J., Van Laarhooven, P. J. M. and Pesch, E. (1991) Genetic Local Search Algorithm for the Travelling Salesman Problem, in *Lecture Notes in Computer Science*, vol 496, 109-116.

[44] Della Croce, F., Tadei, R. and Rolando, R. (1994) Solving a Real World Project Scheduling Problem with a Genetic Approach, in *Belgian Journal of Operations Research, Statistics and Computer Science*, 33(1-2), 65-78.

[45] Della Croce, F., Tadei, R. and Volta, G. (1995) A Genetic Algorithm for the Job Shop Problem, in *Computers and Operations Research*, Jan, **22**(1), 15-24.

[46] Pesch, E. (1993) Machine Learning by Schedule Decomposition, *Working Paper, Faculty of Economics and Business Administration, University of Limburg, Maastricht.*

[47] Mattfeld, D. C. (1996) Evolutionary Search and the Job Shop: Investigations on Genetic

Algorithms for Production Scheduling, in *Physica-Verlag, Heidelberg, Germany*.

[48] Fang, H. L., Ross, P. and Corne, D. (1993) A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling and Open-Shop Scheduling Problems, in *ICGA'5 Proceedings of the 5th International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo, California, pp. 375-382.

[49] J. Adams, E. Balas, and D. Zawack.The Shifting Bottleneck Procedure for Job Shop Scheduling, in *Management Science* 34(3):391-401, 1988.

[50] Rina Dechter and Judea Pearl. Network-Based Heuristics for Constraint Satisfaction Problems, in *Artificial Intelligence* 34(1):1-38, 1988.

[51] N. Raman (1995). Minimum tardiness scheduling in flow shops: Construction and evaluation of alternative solution approaches, in *Journal of Operations Management,* Volume 12, Issue 2, February, Pages 131-151

[52] S. Minton, M.D. Johnston, A.B. Philips, P. Laird. Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method, in *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 17-24. 1990.

[53] Balas, E., and Vazacopoulos, A. (1998) Guided Local Search with Shifting Bottleneck for Job-Shop Scheduling, in *Management Science*, Feb, 44(2), 262- 275.

[54]Barnes, J. W. and Chambers, J. B. (1995) Solving the Job Shop Scheduling Problem Using Tabu Search, in *IIE Transactions*, vol 27, 257-263.

[55]B. Eck and M. Pinedo, Good solution to Job Shop Scheduling Problems Via Tabu Search, Presented at *Joint ORSA/TIMS Meeting, Vancouver, Canada, May 1989*.