2015 International Conference on Soft Computing Techniques and Implementations- (ICSCTI)
Department of ECE, FET, MRIU, Faridabad, India, Oct 8-10, 2015

Genetic algorithm applications on Job Shop Scheduling Problem: A Review

Nisha Bhatt
PG Scholar
Department of Mechanical and Automation Engineering
Indira Gandhi Delhi Technical University for Women,
Delhi, India
bhattnisha09@gmail.com

Nathi Ram Chauhan
Associate Professor& H.O.D
Department of Mechanical and Automation Engineering
Indira Gandhi Delhi Technical University for Women
Delhi, India
nramchauhan@gmail.com

Abstract-Job shop scheduling is an assignment of a number of jobs which are to be processed under a number of machines for certain amount of time. The arrival pattern of the jobs is either known in advance or random. Many approaches like Artificial Intelligence, heuristics, mathematical approaches have been proposed by the researchers to simplify the scheduling tasks. In all the techniques, most of the work is published using Genetic Algorithm (GA). The Job Shop Scheduling Problems (JSSP) ranging from a single machine to flexible JSSP under genetic algorithms have been reviewed. In this paper, a detailed overview of genetic operators and comparison of other techniques with GA have been examined.

Keywords –Job Shop Scheduling; Genetic Algorithms; FMS; crossover; mutation; optimization.

I. INTRODUCTION

Scheduling is an assignment of set of resources to a set of activities over a given amount of time. In production environment, these resources are known as machines and activities as operations. Therefore the task of scheduling is to find a processing order that optimizes the production environment. A job shop is defined as a place where each job is processed by each machine in a predefined order with the constraint of processing one job at a time on the machine. Depending upon the arrival pattern of the jobs, there are static and dynamic scheduling problems. In static, the jobs arrive at the idle shop and must be scheduled. Whereas in dynamic the jobs arrive in random fashion and there is an intermittent arrival of the jobs. Other than this, the sequence of jobs to be processed by the system can be fixed or random. In job shop there are different performance evaluation criteria on which the study is focused in review. Most of the research is focused on minimizing the make span time (total completion time of all the jobs in a schedule)[1] and maximizing the throughput (number of output /jobs per unit time). Job shop scheduling problem belongs to a class of NP-complete problem, such problems are known as 'Hard' problems because as the size of problem increases linearly the computation time increases exponentially. So to solve such a high level of complex production scheduling problems, today most of the study is focus on the use of artificial intelligence techniques, heuristics, meta heuristics in scheduling for example neural network, fuzzy logic, genetic algorithms; particle swarm optimization, simulated annealing etc. This paper reviewed the literature on the use of genetic algorithm in job shop scheduling. The genetic algorithm was developed by John Holland in 1975[28]. However the first paper that enlightened the use of genetic algorithm in job shop scheduling problems was proposed by Davis in 1985 [17]. Since then many researchers have shown their immense contribution in implementing GA in JSSP. Because GA has immersed as a widely used metheuristic in job shop we believe an extensive literature review of the chromosome encoding, selection schemes and genetic operators is needed to assist the researchers to identify their research areas in this field. The purpose of this study is to review the genetic algorithm methodology in JSSP and to provide a comparative study on different approaches (such as neural network, fuzzy logic and shift bottleneck) that have been applied or are currently in demand to solve job shop scheduling problems.

II. A TYPICAL JOB SHOP SCHEDULING PROBLEM

A classical job shop has no flexibility and thus *n* jobs are processed on *m* machines in a classical $n \times m$ job shop. Each job consists of a set of operations where the order of processing is predefined with a single processing plan. The processing time for each operation is fixed for a machine. Usually any scheduling problem is identified by $\alpha|\beta|\gamma$ notations where α signifies the machine environment which is job shop in this case and so is denoted by J. β denotes job characteristics which can be due date, precedence constraints,

release date etc. γ represents optimality condition which can be makespan ,throughput ,tardiness or any weighted combination of these objectives. There are following constraints in job shop:

1. Each operation of a job is to be scheduled after all predecessor operations are completed.
2. Any operation can only be processed by the machine if the machine is ideal.
3. At any time, no two jobs can be simultaneously processed by the machine.

A mathematical formulation of a typical job shop problem is as shown below: [19]:
Index of job *i*
Index of machines *j*
Index of operation *k*
Set of jobs *I= {1…n}*
Set of machines *J= {1…m}*
Set of operations *K= {1…..m}*
It is assumed that number of machines is equal to number of operations.
For any operation *k*, P_k represents set of predecessor operations of a job.
Pt_k is processing time of an operation k.
St_k is starting time of an operation k
B (t) is set of operations to be processed at any time t.
F represents objective function of the problem.
Let $s_{k,j}=1$ if operation *k* requires machine *j* to be processed otherwise $s_{k,j}= 0$
Min F
Subject to

$$St_k - St_j \geq Pt_j \quad k, j=1,2.....m \;\; \&j,k \in P_k \quad (1)$$

$$\sum_{k \in B(t)} s_{k,j} \leq 1 \qquad j \in J \; ; \; t \geq 0 \quad (2)$$

$$St_k \geq 0 \qquad (3)$$

Constraint (1) defines the precedence constraint of operations, constraint (2) defines that a machine can process only one job at a time and constraint (3) represents non negativity of start time of an operation. All these constraints are applicable for minimizing the makespan of the problem however these are valid for any other objective function.

III. GENETIC ALGORITHM

Genetic algorithms are widely used as search techniques based on survival of fittest theory by Darwin. These algorithms are different from other optimization techniques as they do not work directly on task parameters but their coded form. It works on population of points unlike other heuristics that work on one point. It is important to note that GA gives sub-optimal results. The limitation of GA is premature convergence. This problem is solved by using some local search methods such as job insertion or variable neighborhood search. Following are the steps in GA.

i. *Chromosome representation and selection:*

The first step to build or run a genetic algorithm is to provide a suitable encoding for the possible solution for the problem. A possible solution is a string of parameters called genes. This string is called chromosome that represents the solution for the problem being analyzed. Besides the chromosome encoding or representation there is always a fitness function whose value must be evaluated for each encoded solution. In genetic algorithm each chromosome is referred to as an individual. These individuals when grouped together represent a population and during the reproductive phase parents are selected from this population so that they can recombine to produce next generation. In job shops scheduling the following representations and selection methods have been proposed during the last few years.

a) Active schedule representation: An active schedule is one of the feasible schedules where an earlier processing of any operation could delay other operations or could break any precedence constraint. This representation is used to resolve the conflicts among competitive operations on a machine. The selection of competitive operations is based on the idea that no machine is kept idle when it could process some operations. Suppose we have three operations *x, y* and *z* that have to be processed on the same machine. At a time *t* two operations *y* and *z* form the conflicting sets but the third operation *x* does not belong to this set. Here operation *y* can be completed without delaying operation *x* hence *y* is given preference over *z*. Now at a time *t´*operations *x* and *z* form conflicting sets but operation *x* has given preference over *z* because the idle time of machine is the less in this case.[22, 36]

b) *Ordered operation based representation*: In this type of representation chromosome is combination of different genes (α,β) where α denotes jobs to be processed and β denotes machines on which these jobs have to be processed. [33,45]

c) *Matrix representation*: In matrix representation chromosome structure is a two dimensional matrix where chromosome is divided into three segments[35]:

The first segment defines the sequence of order. The second segment represents the assignment of an order to a job and the last segment represents the assignment of a job to a batch. An example of matrix representation is shown in Fig 1

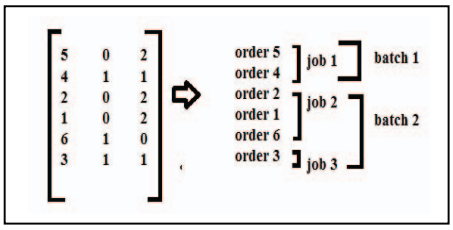

Fig.1: Matrix representation of chromosome

d) *Dominant gene representation* [8,9]: This concept was introduced by Chan et al (2008) for FMS scheduling problems to simplify the identification of dominant gene in chromosome structure. Any genes that undergo slight modification and

results an increase in the fitness value of a chromosome are known as dominant genes. A chromosome is structured in series according to four parameters: (1) machine (2) job (3) operation (4) domination. The length of chromosome string is $\sum n$ where n is summation of total number of machines, jobs and operations. An example of dominant gene representation with 2 machines, 3 jobs and each job possesses two operations is 1231 2131 1210 2230 1310 1221 2110. The first gene 1231 denotes that third operation of second job has to be processed on first machine and this gene is one of the dominant genes of this chromosome structure because the parameter for dominant gene is 1(one) here. The main purpose of dominant gene concept is to keep the potential critical structure and provide them more chance to grow stronger.

ii. Selection scheme:

In GA, parent's selections are based on different selection schemes. Some of them are discussed below.

a) Roulette wheel: It is most famous selection strategy in GA. First of all, calculate the fitness function value of all chromosomes. Let F be the sum of all expected fitness value of chromosomes. Calculate selection probability $p_s=\frac{fitnessvalueofchromosmes}{F}$Now calculate the cumulative probability of all chromosomes and generate a random number R between 0 and 1. Now select the parents according to this number. Repeat this procedure n number of times where n is number of individuals. [37]

b) Tournament approach [62]: Tournament selection is used to evade the effect of premature convergence in genetic algorithms. It uses the relative value of fitness function as criteria to select the individuals from population unlike roulette wheel that is based on proportion of fitness. In this selection process a tournament size has to be defined and then for each individual fitness value of the fitness function has to be calculated so that the best individual can be selected which then move to the mating pool for crossover.

c) Linear ranking method: In linear rank selection method individuals are selected according to the selection probability that depends on their rank. Here first rank is assigned to the worst chromosome (having lowest fitness value) and last rank (equal to total number of individuals) is assigned to the best individual. In order to implement this selection scheme, a random number R between 0 and 1 is generated and then an individual p is selected according to following equation [51]:

$$p = \frac{-2a - b + \sqrt{(2a+b)^2 + 8bR}}{2b}$$

Where a and b are constants. [44]

d) Rank weighted mating strategy: In rank weighted mating method the chromosomes are arranged according to an increasing order of their fitness function value. A probability of selection (pp_k) is being associated with chromosome of rank k which provides a high probability of selection to the chromosomes with poor value of fitness function.

Probability of selection (pp_k) = $\frac{2(n-k+1)}{n(n+1)}$

Where n is total number of chromosomes. [31,24]

Various representation and selection strategy are presented above .For a more thorough coverage on representation (encoding) scheme the reader is referred to the tutorial survey of GA in JSSP that was provided by Cheng et al [14].

iii. Crossover

After the selection of parents, the next step is mating or recombination. In GA parents′ chromosomes are recombined using a mechanism called crossover. This genetic operator is selected in such a way that the formed off-springs have more probabilities of survival than the parents.

a) *Precedence preservative crossover operator (PPX). [5,36,18]:* After selecting the two parents from the chromosome pool, a binary vector of same length as that of parents is formed which represents the order in which operations are selected from the parents. Offspring is formed by drawing leftmost operations (without repetition) from the parents according to the binary vector.

$$parent\ 1 = a\ b\ c\ f\ d\ e$$
$$parent\ 2 = cdfaeb$$
$$binaryvector = 1\ 2\ 1\ 1\ 2\ 2$$
$$offspring = acbfde$$

b) *Random Maximal Preservative Crossover (RMPX) crossover [52,53]:* The following three steps are to be followed for this mating:

- After the selection of parents two crossover points' *p1* and *p2* are chosen randomly.
- Once the crossover points are chosen select an insertion point ix randomly such that it lies in the range [1, n-(*p2-p1*)] where n is length of chromosome.
- Insert the string length [*p1,p2*] of parent 1 in the first offspring from insertion point *ix* and complete the remaining part of first offspring from parent 2.

The second offspring is created by simply reversing the roles of both parents in the crossover with same crossover and insertion points. An illustration of RMPX crossover is as shown below in Fig 2

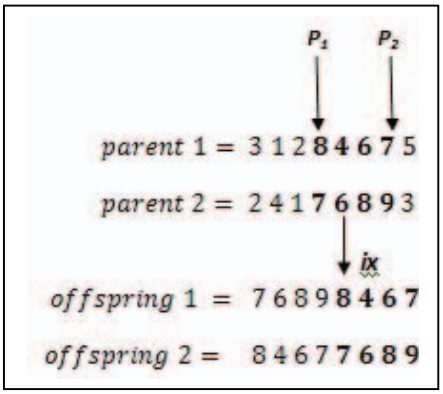

Fig 2: An illustration of RMPX crossover

c) *Two cut point crossover [23, 16,38]:* This is the simplest crossover which involves the selection of two points. The first offspring is created by swapping the middle portion between the two cut points of the parents. Another offspring is created by swapping the portions other than the middle portions of the two parents. An illustration of two cut point operator is as follows:

P1= 100 /1000 /11 *P1= 100 / 1000 / 11*
P2= 001 /1001 / 11 *P2 = 001 /1001 /11*
O1= 100 1001 11 *O1 = 001 1000 11*
O2= 001 1000 11 *O2 = 100 1001 11*

d) *Partially mapped crossover (PMX) [4,7,37*]: To create new chromosomes of the next generation two crossover points are selected randomly and string between these points are exchanged to form new offspring's (same as two point crossover). An example of PMX is as follows:

Parent 1 *1243234132411423*
Parent 2 *2341312413423142*
Child 1 *1243<u>312413</u>411423*
Child 2 *2341<u>234132</u>423142*

To make it in legal form a repair mechanism is applied such that:

Child 1 *1243<u>312413</u>412423*

Similarly,

Child 2 *2341<u>234132</u>423141*

e) *Modified Partially-Mapped Crossover (mPMX):* mPMX operator is a modification in PMX operator to improve the average solution quality. It involves four steps procedure [29]

Step 1: Select two cut points

P1= 123 / 546 /897
P2= 425 / 873 /916

Step 2: Exchange string between two cut points

O1= 123 873 897
O2= 425 546 916

Step 3: Repair the off springs to eliminate duplicity

To repair off springs mapping has to perform in such a way that 8 is replaced by 5, 7 is replaced by 4 and 3 is replaced by 6 in the first offspring. Similarly 5 is replaced by 8, 4 is replaced by 7 and 6 is replaced by 3 in the second offspring.

O1= 126 873 594
O2 = 728 546 913

Step 4: Sort the jobs in the batch according to SPT rule

Let us assume the ascending order of the jobs follows the SPT rule

O1= 126 378 459
O2= 278 456 139

f) *Position based crossover [58,54]:* As the name suggest it depends on the position of selected genes in parents. The genes are selected according to presence of '1' bit on a bitmap string. This bitmap string follows the same length as that of the parents and has randomly generated values '0' or '1' on the positions of a parent. Once the bitmap string is formed the first offspring is created by selecting the genes of first parent according to bitmap string and then carefully scanning the genes of second parent. The non-selected genes of second parent is copied to the first offspring at the same position as that of second parent and the selected genes if also present in second parent are inserted into first offspring according to their relative order in first parent. The second offspring is formed by exchanging the position of two parents and thus follows the same procedure as described above. For example:

P1= 123654 *P2=346125*
Bit map=001011 *Bitmap = 001011*
P2=346125 *P1=123654*
O1=356124 *O2=163254*

g) *Linear order crossover [34, 47]:* In linear order crossover a substring is selected from the first parent and is placed at exact location and with same order (sequence of genes) in an empty offspring. If the genes of that selected substring are also present in second parent then these are deleted and the rest of the genes are copied to the empty offspring by maintaining the order of the second parent.

$$Parent\ 1 = 4\ 7\ 3\ 2\ 8\ 5\ 1$$
$$parent\ 2 = 3\ 6\ 4\ 2\ 1\ 8\ 5$$
$$offspring = \ 6\ 7\ 3\ 2\ 4\ 1\ 8\ 5$$

h) *Uniform order based crossover [50]:* This operator extracts the genes from the first parent at each position with the probability of $pb1 = ff(p1)/[ff(p1) + ff(p2)]$ where *ff (p1) and ff (p2)* are the fitness function value of first parent and second parent respectively. The vacant positions of the child string are filled by gene order as it appears in the second parent. The operator creates the second child by swapping the role of the parents but with the probability of $pb2 = ff(p2)/[ff(p1) + ff(p2)]$.consider the example as shown below:

P1: A B D E C F
P2: C B E A F D
C1: A C D E B F
C2: B D E A F C

i) *Partheno –genetic operator:* To create feasible solution this crossover operator is applied with an adaptive crossover probability *pa*. The first step is to generate a positive integer q randomly. This integer represents the position of q^{th} gene. Now move the genes behind q^{th} gene to the front with remainder genes at the back. Consider an operation based representation with 3 jobs and 3 machines.[57]

Parent chromosome: 5 1 6 3 <u>2 4</u> ; $q = 4$

Offspring: 2 4 5 1 6 3

Adaptive crossover probability

$pa = pa^0(1 + \frac{\gamma(Fav)^\delta}{(Fmax-Fmin)^\delta + Fav^\delta})$ where γ and δ are constants

Pa^0 is initial adaptive crossover probability

Fav is average value of fitness function
Fmax and Fmin are maximum and minimum values of fitness function.

IV. MUTATION

Mutation is another genetic operator that helps GA to not get trapped in local optimization. It seems to be an important operator and must be applied to some individuals to bring population diversity. Different mutation operators have been suggested in past for genetic algorithms in JSSP. Some of them have illustrated below with appropriate examples.

a. Order based mutation (swap mutation or reciprocal exchange mutation)[7]: Two genes were picked up randomly and exchanged during the process of mutation. For example, *c1* is the child form after crossover

$$c1 = 1235422$$

After mutation c1 becomes *c1´*

$$c1' = 1225432$$

b. *Refinement operator [54, 23]:* This operator applies a local search algorithm to identify different sequences and determine the optimal one. According to the algorithm, first the most critical string has to identified that has the major contribution in optimizing the objective function and then a swapping operator is applied within these selected strings. The process is repeated till it improves the solution.

c. *Displacement mutation [39]:* It selects a substring randomly and inserts it into a location that is also selected randomly. All these movements take place within the chosen chromosome. An illustration of displacement mutation is shown below(Fig 3):

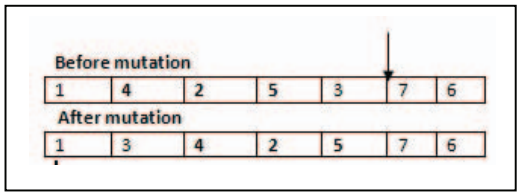

Fig 3: Displacement Mutation

d. *Insertion mutation/ Inversion mutation:* Insertion is a two-step procedure. [41, 61]
Step 1: select a gene randomly
Step 2: insert this gene into a randomly selected location
In inversion, A new chromosome is formed by randomly chosen two points and then inverting the value of genes between them [20, 61, 19]

e. *Shift mutation:* In shift mutation a gene is selected randomly and then shifted either to left or right to a random position from its present locus.

f. *Neighbor mutation operator [13]:* In this mutation, any 'n' numbers of non-identical genes are randomly selected and their possible permutations are calculated. These permutations are then combined with other non-selected genes to obtain possible neighbor of selected chromosome. Let us consider an example:
Suppose we select 3 genes (2, 1, 3) in a chromosome randomly. The possible permutation with these genes is 6.
Selected chromosome: 1 3 2 2 3 1 3 1 2
Neighbor chromosomes: 1 3 2 2 3 3 1 1 2, 1 3 2 1 3 2 3 1 2 , 1 3 2 1 3 3 2 1 2 , 1 3 2 3 3 1 2 12 , 1 3 2 3 3 2 1 1 2

iv. Comparison of GA with other approaches

i. Comparison of artificial neural network (ANN) with GA

Features of ANN:
- Artificial neural networks (NN) are inspired by the central nervous systems of animals.
- An artificial neural network represents the mathematical model of such biological system.
- The models based on NN are capable of processing the information in a parallel distributed fashion [56].

Types of network architectures
1) Hopfield network [21,13]
2) Back-propagation network[10, 49]
3) Competitive network [15]

Limitation of ANN:
- Slow convergence speed
- Long training time

Limitation of GA
- Premature convergence

ANN-GA hybrid features:
1) Good search efficiency
2) No premature convergence
3) Less training time
4) No risk of overdesigning

Many authors have applied the hybrid of ANN and GA in practical applications [42, 59]. For example Haq et al. [27] proposed a hybrid of neural and GA to improve the solution as it would alone obtained by implanting GA. Instead of generating initial population randomly, the solution obtained by ANN is used to provide a starting point for GA. They implied back propagation network architecture in this hybrid where two hidden layers were used with 20 and 30 neurons respectively. Since GA works on a population instead of a single solution, the initial population is a combination of ANN solution and randomly generated population. This hybrid is found effective in lowering the local optima and eliminates the premature convergence of GA. A hybrid of NN and GA acts as an efficient tool to solve the large combinatorial complex problems like JSSP or FMS and helps in designing a small neural network.

ii) Comparison of shifting bottleneck with GA

Features of shifting bottleneck

- In shifting bottleneck procedure, a bottleneck machine is identified by considering the problem as a single machine problem and keeping the other schedules fixed. Thus an algorithm is developed that identify the most critical station in each iteration[2].
- This approach solved the famous 10x10 problem of fisher and Thomson in merely five minutes unlike branch and bound method that took five hours to reach optimality.

Advantage of GA over shifting bottleneck:

- Low computational efforts and time to attain excellent solutions

Limitation of GA

- Memory less algorithms(Lose solutions or previous chromosomes due to unruly effects of genetic crossover).For more information on hybrid of GA and shifting bottleneck readers are referred to survey of hybrid genetic strategies [13].

2) Comparison of fuzzy with GA

- Fuzzy logic is a mathematical technique which incorporates both numerical results from previous solution and past experience. It can incorporate multiple objectives and maps the nonlinear input vector into a scalar output.
- Fuzzy logic uses both numerical and linguistic variables [48].
- These approaches are mostly used in integration with GA or neural networks. In hybrid of GA and fuzzy approach a Fuzzy Logic Controller (FLC) controls the various parameters of genetic algorithm and thus searching ability of GA can be enhanced. In most of the applications the task of FLC is to normalize the ratios of genetic operators. Yun [60]implemented Fuzzy –GA hybrid where two FLC (one for crossover and another for mutation) are applied that take changes in the average fitness of population as their input parameters and use the fuzzy decision table to determine the appropriate control actions. Once the control actions are identified, output is evaluated in terms of changes in mutation and crossover rate with the help of defuzzification table.

V. CONCLUSION

In this article the relevant literature on genetic algorithm in job shop scheduling problems has been surveyed. The following conclusions can be made based on above literature review:

- Various kinds of GA operators are available that strengthen the algorithm and each one are applied to different applications.
- Introducing a knowledge base system that blends personal skills with knowledge provides a means that can enhance the strength of GA. It stores all the selection schemes, crossover and mutation operators that have been applied to the scheduling problems till date and thus automatically selects the most appropriate operators for a particular application[46].
- Another improvement in simple GA can be achieved through constraint based optimization where a knowledge base determines selection schemes and GA operators according to the constraints that have more impact on optimizing a problem [32].
- It can be seen that in the last few years GA have combined with other techniques to form the hybrid algorithm which can enhance the performance of genetic algorithm and provides better solutions.

REFERENCES

[01] A. M. Abazari, M. Solimanpur and H.Sattari,"Optimum loading of machines in a flexible manufacturing system using a mixed-integer linear mathematical programming model and genetic algorithm",Computers & Industrial Engineering, vol. 62, pp. 469–478, 2012.

[02] J. Adams, E. Balas and D. Zawack, "The shifting bottleneck procedure for Job shop scheduling", Management Science, vol. 34, pp. 391-401, 1988.

[03] V. A. Armentano and R. Mazzini, "A genetic algorithm for scheduling on a single machine with set-up times and due dates", Production Planning & Control,vol. 11(7), pp. 713- 720, 2000.

[04] L. Asadzadeh and K. Zamanifar, "An agent-based parallel approach for the job shop scheduling problem with genetic algorithms", Mathematical and Computer modeling, vol. 52, pp.1957-1965,2010.

[05] C. Bierwirth, D.C. Mattfeldand H. Kopfer, "On permutation representations for scheduling problems", in Proceedings of Parallel Problem Solving from Nature IV. Springer, H.M. Voigt, W. Ebeling, I. Rechenberg and H.-P. Schwefel, Eds. Berlin, 1996.

[06] A. Bilgesu and K. Erdem, "A guide for genetic algorithm based on parallel machine scheduling and flexible job-shop scheduling", in Procedia Social and Behavioral Sciences, vol. 62, pp. 817 – 823, 2012.

[07] G. Cavory, R. Dupas and G. Goncalves, "A genetic approach to solving the problem of cyclic job shop scheduling with linear constraints", European Journal of Operational Research, vol. 161, pp. 73–85, 2005.

[08] F. T. S. Chan, S. H. Chung and P. L. Y. Chan, "Application of genetic algorithms with dominant genes in a distributed scheduling problem in flexible manufacturing systems", International Journal of Production Research, vol. 44(3),pp. 523–543, 2006.

[09] F. T. S Chan, S H Chung and L. Y. Chan, "An introduction of dominant genes in genetic algorithm for FMS", International Journal of Production Research, vol.46 (16), pp. 4369–4389, 2008.

[10] Y. Cha, and M. Jung," Satisfaction assessment of multi-objective schedules using neural fuzzy methodology", International Journal of Production Research, 41(8), 1831–1849, 2003.

[11] J. C. Chen, C. C. Wu, C. W. Chen and K. H. Chen, "Flexible job shop scheduling with parallel machines using Genetic Algorithm and Grouping Genetic Algorithm", Expert Systems with Applications, vol. 39, pp. 10016–10021, 2012.

[12] M. Chen and Y. Dong, Y., "Applications of neural networks to solving SMT scheduling problems – A case study" International Journal of Production Research, vol. 37(17), pp. 4007–4020, 1999.

[13] R. Cheng, M. Gen and Y. Tsujimura , "A tutorial survey of job-shop scheduling problems using genetic algorithms", part II: hybrid genetic search strategies, Computers & Industrial Engineering, 36, 343-364, 1999.

[14] R. Cheng, M. Gen and Y. Tsujimura, "A tutorial survey of job shop scheduling problems using genetic algorithms-I representation", Computers & Industrial Engineering, vol. 30(4), pp. 983-997, 1996.

[15] R. M. Chen and Y. M. Huang,"Competitive neural network to solve scheduling problems". Neurocomputing, 37, 177–196, 2001.

[16] S. H.Chen, M. C. Chen and Y. C.Liou, "Artificial chromosomes with genetic algorithm 2 (ACGA2) for single machine scheduling problems with sequence-dependent setup times", Applied Soft Computing, 17,167–175, 2014.

[17] L. Davis, "Job-shop scheduling with genetic algorithms", in Proceedings of international conference on genetic algorithms and their applications, Hillsdale, 1985, pp 136–149.

[18] Y. Demir, and S. K. İşleyen, "An effective genetic algorithm for flexible job-shop scheduling with overlapping in operations", International Journal of Production Research, 2014.

[19] U. Dorndorf and E. Pesch, "Evolution based learning in a job shop scheduling environment", Computers and Operation Research, vol. 22 (1), pp.25-40, 1995.

[20] E. Falkenauer and S. Bouffouix, " A Genetic Algorithm for Job Shop" in Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento, California, 1991, pp. 824-829.

[21] S.Y. Foo, Y. Takefuji and H. Szu," Scaling properties of neural networks for job-shop scheduling". Neurocomputing, vol.8, pp. 79–91, 1995.

[22] B. Giffler and G. Thompson, "Algorithms for solving production scheduling problems". Operations Research, vol. 8, pp.487–503, 1960.

[23] L. D. Giovanni and F. Pezzella, "An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem", European Journal of Operational Research, vol. 200, pp. 395–408, 2010.

[24] D.E. Goldberg, Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison Wesley, 1989.

[25] J. Gu, X. Gu and M. Gu," A novel parallel quantum genetic algorithm for stochastic job shop scheduling", Journal of Mathematical Analysis and Applications, vol. 355,pp. 63–81, 2009.

[26] S.M.K. Hasan, R. Sarker and D. Essam, "Genetic algorithm for job-shop scheduling with machine unavailability and breakdowns", International Journal of Production Research, vol. 49(16), pp. 4999–5015, 2011.

[27] A.N. Haq, T. R. Ramanan, K.S. Shashikant and R. Sridharan, "A hybrid neural network–genetic algorithm approach for permutation flow shop scheduling", International Journal of Production Research, vol. 48(14), pp.4217–4231, 2010.

[28] J. H. Holland, Adaptation in natural and artificial systems: an introductory analysis with applications in biology, control and artificial intelligence, 1975, Ann Arbor: University of Michigan Press.

[29] B. Kim and S. Kim, "Application of genetic algorithms for scheduling batch-discrete production system", Production Planning & Control, 13(2), 155- 165, 2002.

[30] R. Knosala and T. Wal, "A production scheduling problem using genetic algorithm", Journal of Materials Processing Technology, 109, 90-95, 2001.

[31] V. Kodaganallur, A.K. Sen and S. Mitra. Application of graph search and genetic algorithms for the single machine scheduling problem with sequence- dependent setup times and quadratic penalty function of completion times, Computers & Industrial Engineering, 67,10–19, 2014.

[32] A. Kumar, Prakash, M. K. Tiwari, R Shankar and A. Baveja, "Solving machine-loading problem of a flexible manufacturing system with constraint-based genetic algorithm", European Journal of Operational Research, 175, 1043–1069, 2006.

[33] D.M. Lei and H.J. Xiong, "Job shop scheduling with stochastic processing time through genetic algorithm", in Proceedings of International Conference on Machine Learning and Cybernetics, Kunming, China, 2008, pp. 941–964.

[34] C.F. Liaw, , A hybrid genetic algorithm for the open shop scheduling problem, European Journal of Operational Research, vol. 124, pp. 28-42, 2000.

[35] C.H. Liu, "A genetic algorithm based approach for scheduling of jobs containing multiple orders in a three-machine flow shop", International Journal of Production Research,vol.48(15), pp. 4379–4396, 2010.

[36] D. C. Mattfeld and C. Bierwirth, "An efficient genetic algorithm for job shop scheduling with tardiness objectives", European Journal of Operational Research, vol.155,pp. 616–630, 2004.

[37] Moon, S. Lee and H. Bae, "Genetic algorithms for job shop scheduling problems with alternative routings", International Journal of Production Research, vol. 46(10), pp. 2695–2705, 2008.

[38] T. Murata and H. Ishibuchi," Performance evaluation of genetic algorithms for flow-shop scheduling problems", in Proceedings of the First IEEE World Congress on Computational Intelligence, IEEE, 1994, pp. 812–817.

[39] A.C. Nearchou, "The effect of various operators on the genetic search for large scheduling problems" International Journal of Production Economics, vol. 88(2), pp. 191-203, 2004.

[40] I. M. Oliver, D. J. Smith, and J. R. C. Holland, "A Study of Permutation Crossover Operators on the TSP" in Genetic Algorithms and Their Applications: Proceedings of the Second International Conference, J. J. Grefenstette, Ed. : New Jersey, 1987, pp. 224-230.

[41] C. Oguz, and M. F. Ercan, "A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks", Journal of Scheduling, vol. 8, pp. 323–351, 2005.

[42] N. Ozturk, "Use of genetic algorithm to design optimal neural network structure", Engineering Computations, vol. 20(8), pp. 979-997, 2003.

[43] F. Pezzella, G. Morganti and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem" Computers and Operations Research, 35(10), 3202–3212, 2007.

[44] R.K Phanden, A. Jain and R. Verma, "A genetic algorithm approach for flexible job shop scheduling", Applied Mechanics and Materials, vol. 110-116, pp. 3930-3937, 2012.

[45] S. G. Ponnambalam, P. Aravindan, and P. S. Rao, "Comparative evaluation of genetic algorithms for job-shop scheduling", Production Planning & Control, vol. 12(6), pp. 560-574, 2001.

[46] A.Prakash, F. T. S. Chan, and S. G. Deshmukh, "FMS Scheduling with Knowledge Based Genetic Algorithm Approach", Expert Systems with Applications, vol. 38, pp. 3161–3171, 2011.

[47] N. Raeesi, M.R. and Z. Kobti, "A Machine Operation Lists based Memetic Algorithm for Job Shop Scheduling", in IEEE Congress on EvolutionaryComputation(CEC),2011,pp.2436,2443. doi:10.1109/CEC.2011.5949919

[48] S. Rajasekaran and G. A. Vijayalakshmipai, Neural Networks, Fuzzy Logic, and Genetic Algorithms synthesis and applications, 2012, New Delhi: PHI Learning Pvt. Ltd.

[49] I.Sabuncuoglu and S. Touhami, "Simulation metamodelling with neural networks: An experimental investigation", International Journal of Production Research, vol. 40(11), pp. 2483–2505, 2002.

[50] J. Schaller and J .M. S. Valente, A comparison of meta-heuristic Procedures to schedule jobs in a permutation flow shop to minimize total earliness and tardiness,International Journal of Production Research, vol. 51(3), pp. 772–779, 2013.

[51] D. Simon, Evolutionary Optimization Algorithms, Hoboken, 2013 New Jersy: John Wiley & sons.

[52] A. Sioud, M. Gravel and C. Gagne, A hybrid genetic algorithm for the single machine scheduling problem with sequence-dependent setup times, Computers & Operation Research, 39(10), 2415–2424, 2012.

[53] A. Sioud, M. Gravel and C. Gagne, "A genetic algorithm for solving a hybrid flexible flow shop with sequence dependent setup times", in IEEE congress on Evolutionary Computation(CEC), Cancun, Mexico Cancun, Mexico, 2013, pp. 2512-2516, , doi: 10.1109/CEC.2013.6557871

[54] G. Syswerda, "The application of genetic algorithms to resource scheduling" in Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 502- 508, 1991.

[55] W. Teekeng, and A. Thammano, "Modified Genetic Algorithm for Flexible Job-Shop Scheduling Problems", Procedia Computer Science, vol.12, pp.122 – 128, 2012.

[56] J. Wang, W. S. Tang and Roze, C., "Neural Network Applications in Intelligent Manufacturing: An Updated Survey", in Computational intelligence in manufacturing handbook, J. Wang & A. Kusiak,Ed. Boca Raton: CRC Press, 2001, pp. 2.1-2.23.

[57] L. Wang, and D.B. Tang, "An improved adaptive genetic algorithm based on hormone modulation mechanism for job-shop scheduling problem", Expert Systems with Applications, vol.38, pp.7243–7250, 2011.

[58] H. Yu, and W. Liang, “Neural network and genetic algorithm based approach to expand job shop scheduling problem”, Computers & Industrial Engineering, vol. 39,pp. 337-356, 2001.
[59] Y. S. Yun, “Genetic algorithm with fuzzy logic controller for preemptive and non-preemptive job-shop scheduling problems”, Computers & Industrial Engineering, 43, 623–644, 2002.
[60] C. Zhang, Y. Rao and P. Li, “An effective hybrid genetic algorithm for the job shop scheduling problem”, International Journal of Advanced Manufacturing Technology, vol. 39, pp. 965–974, 2008.
[61] G. Zhang, L. Gao and Y. Shi, “An effective genetic algorithm for the flexible job-shop scheduling problem”, Expert Systems with Applications,vol.38,pp.3563–3573,