Global Journal of Computational Intelligence Research.
ISSN 2249-0000 Volume 1, Number 2 (2011), pp. 167-173

http://www.ripublication.com/gjcir.htm

Ways and Means of Applying Genetic Algorithms for Job Shop Scheduling

Shruti Kapoor, Swati Singh, Shelly Chikara, Shivangi Garg and Vijai Singh

Computer Science Department, IMS Engineering College, Ghaziabad, U.P., India
E-mail: shrutikapoor.cse@gmail.com, singh.swati13@yahoo.com, chikara.shelly@yahoo.com, shivi.acse@gmail.com , vijai.cs@gmail.com

Abstract

This paper presents a study of various genetic algorithms developed for solving the Job shop scheduling problem. Various approaches such as crossover operators, mutations and constrained problem statement have been applied to obtain optimal solutions. Some of the key areas studied are: , penalty function, VND[Variable Neighborhood Descent Algorithm], random keys, JOX[job based order crossover], GOX[generalized order crossover] and OBGT[order-based Giffler and Thompson].

Keywords: job shop scheduling, genetic algorithm.

Introduction

The job shop scheduling problem (JSSP) is one of the most well-known optimization problems dealing with the issue of assigning ideal jobs to the available machines at a given time. JSSP is an NP complete problem. The JSSP referred to in this paper can be described as follows[7]:

- Single machine can process only single job at a time
- The processing of a job on a machine is called an operation
- The operations are atomic.
- The consists of at most m operations
- An operations sequence within a job, called machine sequence, and processing time for an operations are given
- An operations sequence on a machine, called job sequence, is unknown. The full sets of job sequences is called a symbolic representation

- A feasible symbolic representations is called a schedule

The aim of jssp is to find a schedule which minimizes the "makespan". The makespan is the total length of the schedule.

Due to the complexity of JSSP, other optimization techniques, such as branch and bound [8] and dynamic programming [9] which are only applicable to modest scale problems fail to obtain good solutions to a large scale problem like JSSP due to the time and space complexity issues. On the other hand, various alternatives of large scale problem solving include heuristic methods, dispatching priority rules, shifting bottleneck approach [10] and Lagrangian relaxation. With the emergence of new techniques from the field of artificial intelligence, much attention has been devoted to meta-heuristics. Two main class of meta-heuristics is the construction and improvement heuristic, such as tabu search [11] and the population based heuristic. Successful examples of population based algorithms include genetic algorithm, artificial immune system and their hybrids, and so on.

Among the above methods, GA, proposed by John Holland [12], uses the basic Darwinian mechanism of "survival of the fittest" and repeatedly utilizes the information contained in the solution population to generate new solutions with better performance. The basic steps involved in GA are:

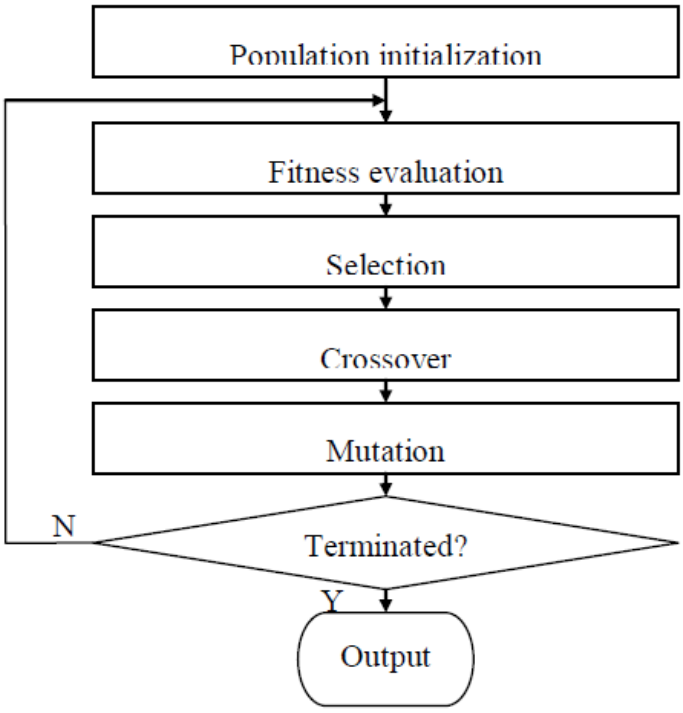


Figure 1. Flow chart of the classical genetic algorithm.

GA is a problem independent approach and is well suited while dealing with hard combinatorial problems.

The penalty method[1] for constraints in JSSP

The penalty function proposed in paper is:

$$P(\vec{x}) = v \cdot k^{\alpha} \cdot (\frac{n_{infeasible}}{n})^{\beta} \cdot \sum_{i=1}^{n_C} w_i < g_i(\vec{x}) >^2$$

where ν is a constant that controls the proportion between $F(\underset{x}{\rightarrow})$ and $P(\underset{x}{\rightarrow})$, k is the iteration number, $n_{infeasible}$ is the number of solutions, n is the number of solutions, $<g_i(x)>$function denotes the degree of violating the *i*-th constraint, is the factor denoting the importance of constraint *i*, and α and β are constants, respectively.

The objective is to minimize the makespan, i.e., the final completion time of all the jobs. For an operation permutation, we can get a complete scheduling by decoding process.

21 best known solutions can be found among the 23 checked instances by using the proposed algorithm, which accounts for 91.3% of the total instances. And the average deviation of the obtained solutions from the best known solution is only around 0.07%.

Generalized order crossover [2]

The GOX operator for permutations with repetition arises from a generalization of OX(Order Crossover). Ox is one of the many permutation representation crossover operators.

The order crossover works as follows:

Step 1: Select a subsection of operation sequence from one parent at random.

Step 2: Produce a proto-child by copying the substring of operation sequence into the corresponding positions.

Step 3: Delete the operations that are already in the substring from the second parent. The resulted sequence of operations contains operations that the proto-child needs.

Step 4: Place the operations into the unfixed positions of the proto-child from left to right according to the order of the sequence in the second parent.

The average solutions of all problems were within the range 0.7% to 7.2% of the best known values whereas the best known solutions only differ by 0% to 4.5%.

The job-based order crossover (JOX) [3]

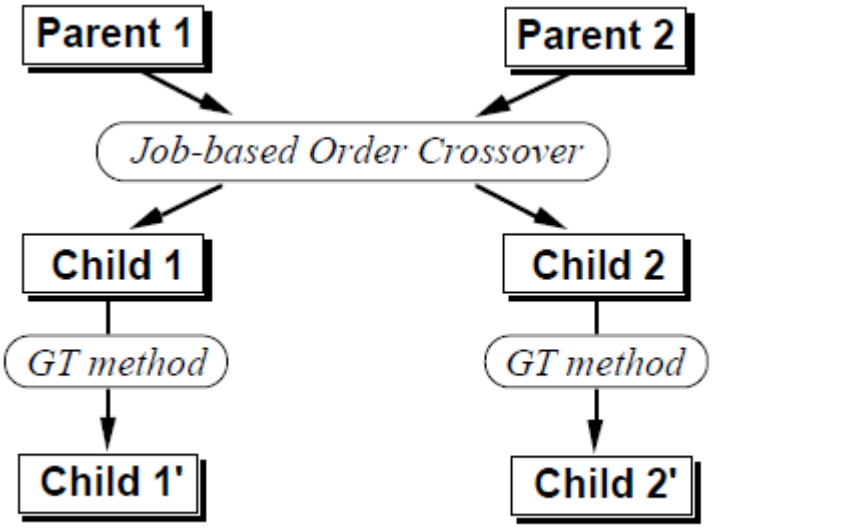
The job-based order crossover (JOX) can preserve characteristics very well. Given job sequence matrixes, parent1 and parent2, JOX generates children, kid1 and kid2, by the following procedure:

1. Randomly, choose the jobs whose locus is preserved.
2. Copy the jobs chosen at step 1 from parent1 to kid1 and from parent2 to kid2,

preserving their locus.

3. Copy the jobs, which are not copied at step 2, from parent2 to kid1 and from parent1 to kid2, preserving their order.

By applying the GT method to the newly generated children, they are modified into active schedules.



Modification by the GT Method after Applying JOX

Hybrid genetic algorithm using random keys [4]

The chromosome representation of the problem is based on random keys. This approach consists in the following three phases:

- *Assignment of priorities and delay times to the operations*. This phase makes use of a genetic algorithm to define and evolve the priorities of the operations and delay times.
- *Construction procedure*. This phase makes use of the priorities and the delay times defined in the first phase, and constructs parameterized active schedules.
- *Local search procedure*. This phase is used to improve the solution obtained by the construction procedure.

The genetic algorithm described in this paper uses a random key alphabet U(0,1) and an evolutionary strategy. Each solution chromosome is made of *2n* genes where *n* is the number of operations.

Chromosome = (genel , gene2 , ..., genen , gene n+1 , ... , gene 2n)

The first *n* genes are used as operations priorities, *Priorityj = Genej*.

The genes between *n+1* and *2n* are used to determine the delay times used when scheduling an operation. The delay time used by each scheduling iteration *g*, *Delayg* ,

is calculated by the following expression:

$$Delay_g = geneg \times 1.5 \times MaxDur,$$

where *MaxDur* is the maximum duration of all operations. The factor 1.5 was obtained after experimental tuning. The HGA obtained the best-known solution for 31 instances, i.e. in 72% of problem instances.

Variable neighborhood descent algorithm for flexible job shop scheduling problems [5]

To strengthen the search ability, individuals of GA are first improved by a variable neighborhood descent (VND), which involves two local search procedures: local search of moving one operation and local search of moving two operations. Let s_{ik} be the starting time of o_{ik} and c_{ik} its completion time. An operation o_{ik} can only be started after its immediate job predecessor $o_{i(k-1)}$ is completed. Given a time interval $[tS_j, tE]$ beginning from tS_j and ending at tE_j on machine j to allocate o_{ik}, we have, $S_{ik} = \max\{tS_j, c_{i(k-1)}\}$ if $k \_ 2, tS_j$ if $k = 1$.

Time interval $[tS_j, tE_j]$ is available for o_{ik} if there is enough time span from the starting of o_{ik} until the ending of theinterval to complete it, $\max\{tS_j, c_{i(k-1)}\} + p_{ikj} \_ tE_j$ if $k \_ 2, tS_j + p_{ikj} \_ tE_j$ if $k = 1$.

The evolution speed of simple GAs is relatively slow [21]. One promising approach for improving the convergence speed to improved solutions is the use of local search in GAs. Within a hybrid of GA and local search, a local optimizer is added to GAs and applied to every child before it is inserted into the population. Variable neighborhood search (VNS) is based on a simple principle: systematic change of neighborhood within a possibly randomized local search [23,24]. The VND method is obtained if change of neighborhoods is performed in a deterministic way.

They got the same best solution as that obtained by the combined efforts of previous works for 119 instances, and found 38 new better solutions.

Order-based Griffer and Thompson algorithm [6]

Order-based Griffer and Thompson algorithm has the following characteristics:

- A direct chromosome representation, containing the schedule itself is used. This includes the permutation of jobs for each machine. Each operation contains its start time
- Order based operators were implemented
- The GT and ND algorithms were utilized to generate active and non delay schedules
- Hillclimb each offspring with the Gravowsky critical neighborhood
- A selection and replacement strategy similar to GENITOR is used. There is no selection of parents. The members of population get a chance to reproduce in turn (going from best to worse), with a crossover probability Pc. A mate is chosen randomly

Comparison of Algorithms

	Selection	Crossover	Mutation	Fitness criteria	Infeasible solution	Characteristic
The penalty method[1] for constraints in JSSP	Random	Normal	Hyper-mutation	High threshold value to replace k worst chromosomes by k best	Taken	Constrained JSSP using Penalty function
Generalized order crossover [2]	Neighborhood mating and local offspring acceptance	GOX i.e. permutation with repetition	Position based mutation	Active scheduling	Not taken	Generalized order crossover
The job-based order crossover (JOX) [3]	Random	JOX	Shift change	Ranking method	Taken(transformation from infeasible to active)	Static JSSP and Job based order crossover
Hybrid genetic algorithm using random keys [4]	Random and top 10% direct	Parameterized uniform crossover	Replacing bottom 20% by random generation	Minimum makespan function	Not taken	Priority of operation and delay time
Variable neighborhood descent algorithm for flexible job shop scheduling problems [5]	Ranking based	Order-based	Allele based and immigration based	Roulette wheel selection	Not taken	VND, local search space optimization, flexible JSSP
Order-based Griffer and Thompson algorithm [6]	50%ND+ 50%GT	Order-based	Sublist permutation using order-based	Fitness value taken	Not taken	Static JSSP

Conclusion

Based on our comparison done in the above section, we conclude that there is a tradeoff between optimal solutions and computation time. Since JSSP is a NP complete problem, there exists no specific algorithm which would give the optimal solution for all cases. We observed that, whenever a constrained JSSP was taken, the solutions were near optimal than the results obtained by modified crossover techniques.

References

[1] Liang Sun , Xiaochun Cheng, Yanchun Liang," Solving Job Shop Scheduling Problem Using Genetic Algorithm with Penalty Function" International Journal of Intelligent Information Processing Volume 1, Number 2, December 2010

[2] Christian Bierwirth in his paper "A Generalized Permutation Approach to Job Shop Scheduling Problem with Genetic Algorithms"

[3] Isao Ono Masayuki Yamamura Shigenobu Kobayashi in their paper, "A Genetic Algorithm for Job-shop Scheduling Problems Using Job-based Order Crossover" Proc. of ICEC'96, pp.547-552 (1996)

[4] José Fernando Gonçalves, Jorge José de Magalhães Mendes, Maurício G.C. Resende in their paper "A Hybrid Genetic Algorithm for the Job Shop Scheduling Problem"

[5] Jie Gao, Linyan Sun, Mitsuo Gen in their paper, "A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems" Computers & Operations Research 35 (2008) 2892 – 2907

[6] Manuel Vazquez in his paper," A comparison of genetic algorithms for job shop scheduling" conventional genetic algorithm for job shop scheduling by ryohei nakano and takeshi yamada

[7] P. Brucker, B. Jurisch, B. Sievers, "A branch and bound algorithm for job shop scheduling problem", Discrete Applied Math, vol. 49, pp. 105-127, 1994.

[8] T. Lorigeon, "A dynamic programming algorithm for scheduling jobs in a two-machine open shop with an availability constraint", Journal of the Operational Research Society, vol. 53, no. 11, pp. 1239-1246, 2002.

[9] J. Gao, M. Gen, L. Y. Sun, "A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems", Computers and Industrial Engineering, vol. 53, no. 1, pp. 149-162, 2007.

[10] C. Y. Zhang, P. G. Li, Y. Q. Rao, "A very fast TS/SA algorithm for the job shop scheduling problem", Computers & Operations Research, vol. 35, pp. 282-294, 2008.

[11] J. H. Holland, Adaptation in Natural and Artificial Systems. Ann Arbor, MI: University of Michigan Press, 1975.