Extremal Optimization for Solving Job Shop Scheduling Problem

Masoud Gharehjanloo
Software Engineering Department, Minoodasht Branch-Islamic Azad University, Minoodasht, Iran
gharehjanloo@yahoo.com

Majid Vafaei Jahan
Software Engineering Department, Mashhad Branch -Islamic Azad University, Mashhad, Iran
VafaeiJahan@mshdiau.ac.ir

Mohammad-R. Akbarzadeh-T.
Departments Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.
Akbarzadeh@ieee.org

Masoud Nosratabadi
Software Engineering Department, Mashhad Branch-Islamic Azad University, Mashhad, Iran
msd.nosratabadi@gmail.com

***Abstract*—Job shop is one of a well known NP-hard optimization problems. In this paper, extremal optimization is proposed for job shop scheduling. Extremal optimization is an evolutionary meta-heuristic method that consecutively substitutes undesirable variables in current solution with a random value and evolves itself toward optimal solution. For EO, the quality of generated initial solution plays an important role in convergence rate and reaching global optimum; hence GT method is utilized for initial solution. This algorithm is implemented on several sample problems on LA datasets and show that optimal solution can be reached quickly on most of the datasets.**

Keywords — Extremal optimization, job shop scheduling problem, GT algorithm, local fitness

I. INTRODUCTION

Job shop scheduling problem (JSSP) is the most common type of scheduling problems and is one of the hardest combinatorial problems which has been considered in recent decades by many researchers. A job shop scheduling problem is determined by a finite set J consisting of n jobs, $J = \{J_1, \dots, J_n\}$, which has to be scheduled on a finite set M including m machines, $\mathrm{M} = \{\mathrm{M}_1, \dots, \mathrm{M}_\mathrm{m}\}$. Each job J_j is fragmented into a series of m operations o_{ik}, where k represents machine M_k which operation i has to be processed on.

Technical order of machines for each job J_j (processing route) was determined before. Every operation o_{ik} has its own processing time p_{ik}, that is shown by a non-negative number. General form of job shop scheduling problem consists of the following constraints[1]: 1) a job is processed only once on each machine. 2) a job is processed according to a specific order on machines. 3) all jobs are accessible at time zero. 4) each machine processes just one job at a time and operations cannot stop. 5) a job can not be executed simultaneously on several machines.

The problem goal is to find a schedule to minimize the makespan (C_{max}), that is, the time required to complete all jobs. Garey et al.[2] demonstrated that JSSP is an NP-hard problem, so it is unsolvable in a reasonable computational time. So far, different methods have been presented for solving JSSP such as simulated annealing[3-5], shifting bottleneck algorithm[6-7], Tabu search[8], particle swarm optimization[9-10], genetic algorithm[11] and artificial immune system[12].

Extremal optimization is one of the evolutionary methods which has been executed successfully on some optimization problems[13]. To solve job shop scheduling problem, extremal optimization method is proposed. Extremal optimization operates only on one individual and at each iteration improves only one individual.

In the following, Section II introduces extremal optimization and the proposed structure describes in Section III. In Section IV, experimental results is studied. At last, Section V is discuss conclusion.

II. EXTREMAL OPTIMIZATION ALGORITHM

Extremal optimization is a local-search heuristic algorithm that was presented by Boettcher[14]. This algorithm is general purpose algorithm for finding the most qualified solution for hard problems and that has been successfully applied on some of NP-hard combinatorial optimization problems[13, 15]. This method (EO) is consecutively substituting undesirable variables with a random value in solution near optimal point.

Unlike genetic algorithm that works on a population of chromosomes, EO evolves a single individual(S). In EO, every decision variable in the current individual S is considered as a "species" and only mutation operator exists and always mutation is done on worse species and specifically can improve its elements and moves toward optimal solution generation by generation. For this, it is necessary to relate a quality amount (i.e. fitness) to each species (a local fitness is related to each element of solution that determines the effectiveness of that element in total fitness); that, this is the discrimination of this method with holistic methods such as evolutionary algorithms that relate the same fitness to all species of solution based on their population evolution versus an objective function. The most obvious difference of extremal optimization algorithm and other heuristic algorithms such as genetic algorithm is the algorithm's need to know the species fitness (local fitness) in addition to global fitness of solution. General structure of EO algorithm for a minimization problem with n decision variables is as following:

1. A individual S is generated randomly. The best

solution is adjusted to
$S_{best} = S$.
2. For the current individual S
a) For each decision variable x_i, $i \in \{1, \dots, n\}$, local fitness λ_i is determined.
b) j is found such that $\lambda_j \leq \lambda_i$ for all i's to be satisfied. That is, x_j has the worst fitness.
c) A $S' \in N(S)$ is selected in a way that x_j has to change its form, $N(S)$ is the neighbor of S.
d) $S = S'$ is accepted unconditionally.
e) If current cost function value is less than optimal cost function value, i.e.
$C(S) < C(S_{best})$, then $S_{best} = S$.
3. Step 2 is iterated as much as desirable.
4. S_{best} and $C(S_{best})$ is given as output.

Basic algorithm for optimization problems is very competitive when EO is able to select randomly its choice among many neighborhoods ($S'\epsilon$N(S)) that satisfy step (2c). however, sometimes selecting neighborhood N for the problem faces EO to a certain process: always, choosing the worst variable in step (2c) gives no option for step (2c). Like iterative improvement, by this method, an EO process is trapped into local process. To avoid getting trapped in local optimum and for global improvement of results, Boettcher et al. introduced a single parameter for this algorithm[13] and the new algorithm is called $\tau - \mathrm{EO}$. This parameter τ, remains constant during every iteration and for each problem is different with the size of system(n).the new a Parameter τ permits for utilizing from kept memory in ordered fitness for x_i with more details. A permutation Π of labels i is found with

$$\lambda_{\Pi(1)} \leq \lambda_{\Pi(2)} \leq \cdots \leq \lambda_{\Pi(n)} \qquad (1)$$

That is, species are ordered increasingly according to their fitness such that the worst variable x_j (step (2b)) is placed in rank 1, $j = \Pi(1)$, and the best variable is placed in rank n. now, a probability distribution is considered on rank k:

$$P_k \propto k^{-\tau} \quad , \quad 1 \leq k \leq n \qquad (2)$$

That is, a probability is given to each species. The difference of this new algorithm and basic extremal optimization algorithm is related to selection step of replacement species. In this algorithm, for finding replacement species, first, species are ordered increasingly with respect to fitness, then a value P_k, is assigned to each species that is computable by (2) where k is the species order in the ordered list. In every update, a rank k is selected according to P_k.

III. THE PROPOSED ALGORITHM

Extremal optimization algorithm discussed in section II has been used to find the worst job and its mutation. if instead of using random solutions, a set of qualified solutions is used as the initial solution, it will affect so much on convergence rate and efficiency of algorithm. Therefore, initial individual is generated by means of Giffler-Thompson algorithm that will be described in section III.B Schedules that are generated by this algorithm are of active scheduling type.

$M_1: (J_1, J_3, J_2)$
$M_2: (J_2, J_3, J_1)$
$M_3: (J_1, J_2, J_3)$

Figure 1- a) a complete sample of solution structure

M_1:	1	3	2
M_2:	2	3	1
M_3:	1	2	3

Figure 1- b) summarized sample of solution structure

A. solution structure

Selection of solution structure in JSSP is of great importance that can affect on algorithm speed, consuming power and convergence rate. In this paper, job sequence matrix representation so-called "permutation representation" has been used. In this method, an $m \times n$ matrix is defined (m is the number of machines and n is the number of jobs) where rows of this matrix denote machines and columns denote the job process order. In this method, every machine has its own priority list. An example of this structure is observed in Fig. 1 where in Fig. 1.a, its complete structure and in Fig. 1.b a summarized structure is shown.

B. Initial individual

In evolutionary algorithms and specially, in JSSP, initial solutions have considerable influence in convergence rate and final solution quality. Thereafter, for generating initial individual from Giffler-Thompson [16] so-called GT has been used. This algorithm by adding operations successively generates active and feasible scheduling. Also, it was used for decoding the solution structure and repairing it. Symbols applied in this algorithm are as following:

Q	The set of currently schedulable operations.
first (i)	The first operation of job J_i
$succ^T(o)$	The set of technological successors of operation o
$s^e(o)$	The earliest possible starting time of operation o
$c^e(o)$	The earliest possible completion time of operation o, i.e. $s^e(o) + p(o)$.
K	The conflict set
$\acute{Q}$	The set of schedulable operations on the machine which has just been occupied.
$m(o)$	The machine which operation o is assigned to.
$\hat{r}_k$	The earliest time when machine k gets idle again.

GT algorithm is shown in algorithm 1.

Algorithm 1: The Giffler-Thompson (GT) algorithm for job shop scheduling

$Initialize\ \ Q \leftarrow \cup_{i=1}^{n} \{first(i)\};$
$s^e(o) \leftarrow 0, \forall_o \epsilon Q;$
$\hat{r}(k) \leftarrow 0, \forall\ 1 \le k \le m;$
while $Q \neq \emptyset$ *do*
  $Determine\ c_{min} = min_{o \epsilon Q}\left(s^e(o) + p(o)\right);$
  $Build\ conflict\ set$
    $K = \{o | o \epsilon Q \wedge m(o) = m(o_{min}) \wedge s^e(o) < c_{min}\}$
  $Choose\ operation\ \acute{o} \epsilon K\ to\ be\ scheduled\ next;$
  $Remove\ \acute{o}\ from\ Q: Q \leftarrow Q \backslash \{\acute{o}\};$
  $Set\ \hat{r}\left(m(\acute{o})\right) = c^e(\acute{o});$
  $Determine\ set\ Q = \{o | o \epsilon Q \wedge \acute{m}(o) = m(\acute{o})\};$
  forall $o \epsilon \acute{Q}$ **do**
    $s^e(o) \leftarrow \max\left(s^e(o), \hat{r}\left(m(\acute{o})\right)\right);$
  end
  forall $o \epsilon succ^T(\acute{o})$ **do**
    $Q \leftarrow Q \cup \{o\};$
    $s^e(o) \leftarrow \max\left(s^e(o), \hat{r}\left(m(o)\right)\right);$
  end
end

C. Local fitness

In order to solve JSSP with extremal optimization algorithm, a local fitness should be defined for each job. Since, in job shop scheduling problem, if jobs with more processing time complete sooner, the possibility of decreasing makespan (C_{max}) increases. For this, more priority should be considered for jobs with more processing time. In this paper, in order to relate a local fitness and to compute it, the processing time of a job is subtracted from duration has lasted to complete this job by itself (from the time when its first operation to be assigned to a machine until the time when the last operation of that job to be completed) and the result is multiplied by its processing time.

This local fitness function can be expressed in another way: sum of gaps exist between successive operations of a job to be calculated and multiplied by its processing time. Multiplication was performed because jobs with more processing time to have more priority.

λ_i= *((starting time of the first operation of J -completion time of the last operation of J)-total time needed for job J)*×*total time needed for job J*

In above formula, "total time needed for job J" is the minimum time needed for job J to complete its processing. For example, a scheduling problem with three jobs is considered which jobs processing order represented in Table 1 and processing time of each job is represented in Table 2.

Table 1: jobs processing order

J_1: (3,1,2)
J_2: (2,3,1)
J_3: (2,1,3)

Table 2: jobs processing time

	M_1	M_2	M_3
J_1	40	20	70
J_2	30	50	60
J_3	20	40	30

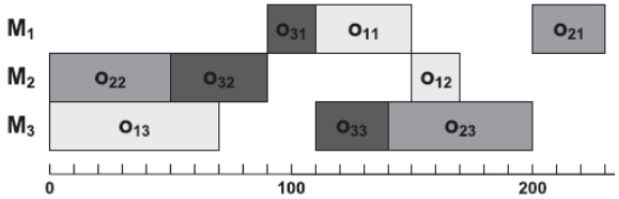

Figure 2. initial schedule for example problem

It is supposed an initial schedule was generated for this problem like Fig. 2. The goal is to compute local fitness for job J_2. As it is observed in Fig. 2, the first operation of job J_2 (i.e. operation o_{22}) starts at time zero and the last operation of job J_2 (i.e. operation o_{21}) ends at time 230. Also, total time needed for job J_2 (i.e. sum of times of all operation of job J_2) is 140. Therefore, local fitness for job J_2 is as following:

$$\lambda_{J_2} = \left((230 - 0) - 140\right) * 140 = 12600$$

Also, local fitness of job J_1 and J_2 is computed as:

$$\lambda_{J_1} = \left((170 - 0) - 130\right) * 130 = 5200$$
$$\lambda_{J_3} = \left((140 - 50) - 90\right) * 90 = 0$$

As it is observed, job J_2 has the largest local fitness and is known as the worst species that mutation operation has to be done on.

D. proposed structure

The proposed algorithm consists of following steps:

step1 initial individual is generated by *GT* algorithm. In this step, the fitness of individual is calculated at the same time of its generating.

step2 computing the local fitness of every job and determining the worst.

step3 a neighborhood is generated for individual with respect to the worst determined job. In generating the neighborhood of the worst recognized job in step 2 and, mutation happens it is done as every operation related to this job in its corresponding machine and is swap with another job or inserted to another location.after mutation, new solution structure is disturbed and generated schedule by solution is not feasible. To make an feasible scheduling according to new solution, *GT* algorithm is imposed one more time on new solution. After

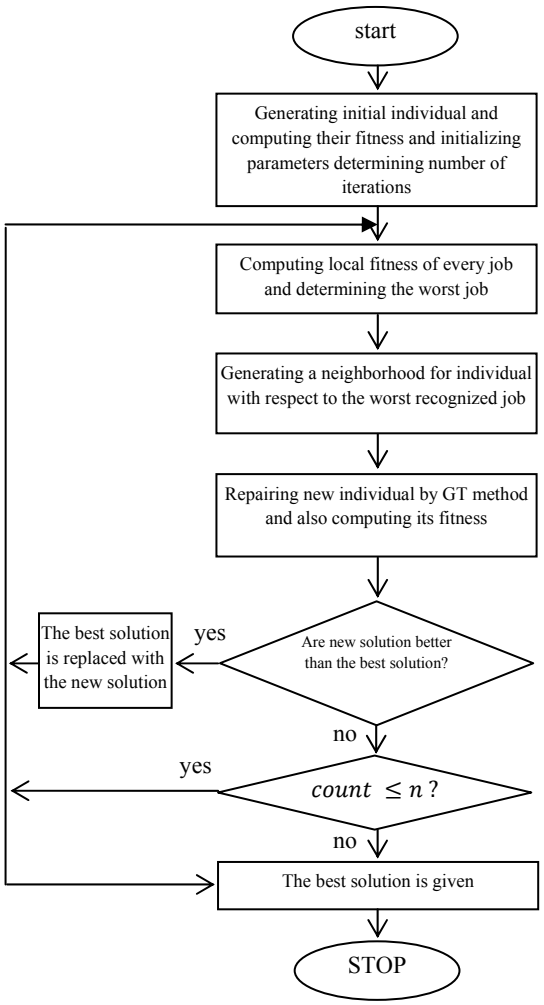

Figure 3: the proposed algorithm flowchart

step4 operating *GT* algorithm, the new solution generates an feasible scheduling which is active too. Like step1, during operating *GT* algorithm, the new solution fitness is also computed.

step5 if new solution is better than the best solution, the best solution is replaced with new solution. this algorithm repeated until $n <= count$.

step6 the best value is given.

In order to understand better this method, its flowchart is shown in Fig. 3.

IV. EXPERIMENTAL RESULTS

To show the effectiveness of the proposed algorithm, it was tested on some data sets of *LA* which were designed by Lawrence, 1984. These datasets were selected such that problems with different sizes to be taken into account. It was implemented by MATLAB software on a computer with Intel processor 2.26 GHz. The parameter of the τ were considered 1.6. For example, the proposed algorithm behavior for problem LA39 is shown in Fig. 4

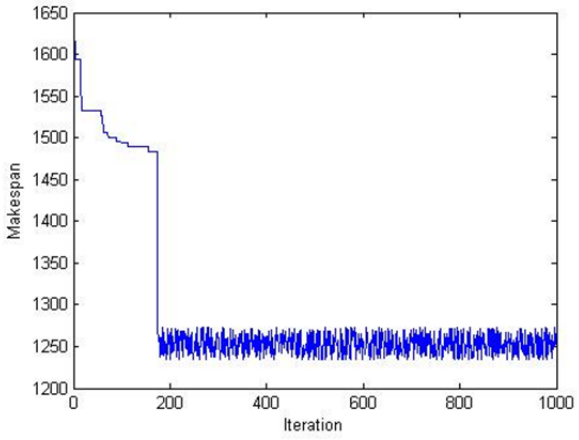

Figure 4: the proposed structure behavior for LA39

Since, EO uses only one mutation operator, so is of higher convergence rate in contrast to other meta-heuristic algorithms like genetic algorithm, particle swarm optimization and other population based meta-heuristic methods.

Results of this simulation are shown for mentioned data sets in Table 3. The third column of Table 3 determined with *BKS* (best known solution) show the best found solution for mentioned problems. For every data set, the proposed algorithm was executed 20 times on it, which the resulted average value during 20 iterations is given in Table 3. For comparison, three samples of the best done jobs in job shop field are represented in Table 3.

V. CONCLUSION

In this paper, extremal optimization method is proposed for job shop scheduling problem. As it was observed, what makes EO superior than other meta-heuristic methods are its high convergence rate and more efficiency and also less complexity and easy implementation. The algorithm is tested on a set of 8 standard instances of LA dataset, and the results are compared with those obtained using other existing approaches. These results indicate that the proposed algorithm is an attractive alternative for solving the job shop scheduling problem.

Table 3. computational results for sample problems

Problem	Size	BKS	TS-SB	HGA-Param	PSO-GT	EO				
	$m \times n$		Pezzella & Merelli[8]	Gonçalves et al [11]	Sha & Hsu [9]	C_{max}	Average	Best solution time(s)	Total time(s)	Precision (%)
LA01	10×5	666	666	666	666	666	55.0	0.4	43.9	100
LA06	15×5	926	926	926	926	926	926.0	0.5	56.7	100
LA11	20×5	1222	1222	1222	1222	1222	1222.0	2.7	64.5	100
LA20	10×10	902	902	907	902	907	913.3	10.3	87.3	99.4
LA22	15×10	927	927	935	927	935	960.3	45	221.5	99.4
LA28	20×10	1216	1216	1232	1216	1216	1241.7	53.5	496.6	100
LA31	30×10	1784	1784	1784	1784	1784	1784.0	23.3	903.8	100
LA39	15×15	1233	1240	1246	1233	1233	1276.6	73.2	305.2	100

References

[1] French, S., Sequencing and scheduling: An introduction to the mathematics of the job shop. 1982: UK: Horwood.

[2] Garey, M.R., D.S. Johnson, and R. Sethi, COMPLEXITY OF FLOWSHOP AND JOBSHOP SCHEDULING. Mathematics of Operations Research, 1976. 1(2): p. 117-129.

[3] Steinhöfel, K., A. Albrecht, and C.K. Wong, Two simulated annealing-based heuristics for the job shop scheduling problem. European Journal of Operational Research, 1999. 118(3): p. 524-548.

[4] Steinhöfel, K., A. Albrecht, and C.K. Wong, An experimental analysis of local minima to improve neighbourhood search. Computers & Operations Research, 2003. 30(14): p. 2157-2173.

[5] Zhang, R. and C. Wu, A simulated annealing algorithm based on block properties for the job shop scheduling problem with total weighted tardinessobjective. Computers & Operations Research, 2011. 38(5): p. 854-867.

[6] Adams, J., E. Balas, and D. Zawack, The shifting bottleneck procedure for job shop scheduling. Manage. Sci., 1988. 34(3): p. 391-401.

[7] Balas, E. and A. Vazacopoulos, Guided Local Search with Shifting Bottleneck for Job Shop Scheduling. Manage. Sci., 1998. 44(2): p. 262-275.

[8] Pezzella, F. and E. Merelli, A tabu search method guided by shifting bottleneck for the job shop scheduling problem. European Journal of Operational Research, 2000. 120(2): p. 297-310.

[9] Sha, D.Y. and C.-Y. Hsu, A hybrid particle swarm optimization for job shop scheduling problem. Computers & Industrial Engineering, 2006. 51(4): p. 791-808.

[10] Zhang, H., et al., Particle swarm optimization-based schemes for resource-constrained project scheduling. Automation in Construction, 2005. 14(3): p. 393-404.

[11] Gonçalves, J.F., J.J. de Magalhães Mendes, and M.G.C. Resende, A hybrid genetic algorithm for the job shop scheduling problem. European Journal of Operational Research, 2005. 167(1): p. 77-95.

[12] Zhang, R. and C. Wu, A hybrid immune simulated annealing algorithm for the job shop scheduling problem. Applied Soft Computing, 2010. 10(1): p. 79-89.

[13] Boettcher, S. and A.G. Percus, Extremal optimization: An evolutionary local-search algorithm. Computational Modeling and Problem Solving in the Networked World, 2002. 21: p. 61-77.

[14] Boettcher, S., Optimizing partitions of percolating graphs. Physica A: Statistical Mechanics and its Applications, 1999. 266(1-4): p. 100-103.

[15] Boettcher, S. and A.G. Percus, Extremal optimization at the phase transition of the three-coloring problem. Physical Review E, 2004. 69(6).

[16] Giffler, J., & Thompson, G. L, Algorithms for solving production scheduling problems. Operations Research, 1960. 8: p. 533–549.