A genetic based hyper-heuristic algorithm for the job shop scheduling problem

Jin Yan
Department of logistics engineering, School of Mechanical Engineering, University of science and Technology Beijing,
Beijing, China
e-mail: yanjin2009_2012@126.com

Xiuli Wu
Department of logistics engineering, School of Mechanical Engineering, University of Science and Technology Beijing,
Beijing, China
e-mail: wuxiuli@ustb.edu.cn

***Abstract*—Job shop scheduling problem (JSP) is a typically NP-hard problem. In this paper, we proposed a genetic based hyper-heuristic algorithm for solving JSP. The hyper-heuristic algorithm employs two level frameworks. In the high-level, genetic algorithm is employed to explore the whole solution space; while in the low level, there are 16 combinations coming from 4 frequently used crossover operators and 4 mutation operators. Time window is introduced to learn from the recent searching process. Finally, the result on the benchmark instance FT10 shows that the time window helps the proposed algorithm to search effectively. Furthermore, to verify the validity of the proposed algorithm, some traditional genetic algorithms with fixed genetic operators are also employed to test FT10, supporting the statement that it is much better using adaptive operator selection in GA than using a fixed operator.**

Keywords-job shop scheduling problem; genetic algorithm; hyper-heuristics; adaptive operator selection; time window

I. INTRODUCTION

Job shop scheduling problem (JSP) is a typical NP-hard combinatorial optimization problem, which is difficult to obtain the optimal solution in polynomial time. Many scholars have carried out a lot of research on JSP and proposed many methods, such as the branch and bound method [1], mathematical programming methods [2], local search [3], heuristic rules [4], artificial intelligence [5], et al. Using genetic algorithm solving JSP has been the focus of researchers [6~7].

Genetic algorithm inspired by the "survival of the fittest" of Darwin, is an iteration process that helping obtain the fittest individual to adapt to the environment by applying selection, crossover and mutation. Crossover inherits genes from parents and mutation increases the diversity of the population to avoid local prematurity. In general, crossover operator and mutation operator in traditional genetic algorithm are predefined. However, it is difficult to determine in advance which or what kinds of operators are most appropriate for a specific problem. The human expert can only adjust genetic operator continuously through the results calculated by simulation in offline state. Human expert needs to do a lot of design works, which are time-consuming and difficult to guarantee to find the most suitable operator, also greatly limit the use in the ordinary users. As a result, finding an adaptive algorithm adjusting the parameters automatically has become a focus in the field of evolutionary algorithms recently. Hyper-heuristics algorithm, therefore, emerged to adaptively select operators.

Hyper-heuristics are multi-level/multi-layer algorithms that manage a set of heuristic operators or generate low-level heuristics from existing operators [8]. Hyper-heuristics search the space of heuristics rather than that of solutions, and use limited problem specific information to control the search process. The problem specific information is encapsulated into the problem model and a pool of low-level heuristics or search operators [9]. As a low-level heuristic could be considered already known heuristic operators, simple metaheuristic algorithms or even a group of other hyper-heuristics. Each low-level heuristic can search the solution space, modify the current solution and find a new one [10].

This paper combines hyper-heuristics with genetic algorithm to solve job shop scheduling problem. This method is able to adaptively select the best genetic operator in each iteration.

II. THE JOB SHOP SCHEDULING PROBLEM

The job shop scheduling problem composes of n jobs and m machines. Each job composed of m operations that must be processed on m machines. Each operation occupies one of the m machines for a fixed duration. Each machine can process at most one operation at a time and the processing of an operator cannot be interrupted once started. The operations of a given job have to be processed in a given order. The aim is to find a schedule of the operations on the machines, to obtain the start time and end time of each operation, and to minimize the makespan (C_{max}).

LetC_{ij}, $t_{ij}$and p_{ij} represent the finish time, start time and processing time of operation (i, j), respectively. The pair (i, j) refers to the operation of job j on machine i. Job j is processed on machine i before it is processed on machine s.

The optimization model of the JSP can be described as follows:

$$min\ C_{max} = min\left(max\left(C_{ij}\right)\right) \quad (1)$$

Subject to

$$t_{ij} \geq 0 \quad (2)$$

$$t_{ij} - t_{ik} \geq p_{ik} \ \ or \ \ t_{ik} - t_{ij} \geq p_{ij} \quad (3)$$

$$t_{sj} - t_{ij} \geq p_{ij} \quad (4)$$

978-1-4799-8646-0/15 $31.00 © 2015 IEEE
DOI 10.1109/IHMSC.2015.13

$$j,\ k=1,2,\cdots,n,\ i,\ s=1,2,\cdots,m \tag{5}$$

The objective function (1) minimizes the makespan. Constraint (2) means that all jobs can be released at time 0 and ensures the finish times of all operations are non-negative. Constraint (3) states that one machine can only process one operation at a time. Constraint (4) represents the precedence relations among the operations of the same job. Constraint (5) represents the range of variables.

III. THE GENETIC BASED HYPER-HEURISTIC ALGORITHM

A. The high-level design

Genetic algorithm is a general search technique based on genetic process of biological organisms, aims at optimizing some objective function, referred to as fitness function.

Before a GA can be run, a suitable encoding (or representation) for the problem must be devised and a fitness function is required, which assigns a figure of merit to each encoded solution. During the run, the genetic algorithm evolves a population of individuals by iteratively (1) selecting some individuals (the parents) from the old population; (2) perturbing stochastically the parents using crossover or mutation operator, thus generating offspring.

In this paper, the traditional genetic algorithm is employed as the high-level algorithm where encoding, evaluation, selection, crossover and mutation are still the main operations. Different from that in the traditional way, the crossover operator and mutation operator are treated as a combination rather than a single operator. An adaptive operator selection (AOS) scheme is employed to choose the provided combinations.

1) Encoding

In GA, chromosomes represent solutions of the concerned problems. Thus, an encoding method is needed to convert the concerned problem to a chromosome.

Using Operation based representation [11], each chromosome consists of $n\times m$ natural number component and each job appears m times**.** The operation number is determined by the frequency the corresponding job appears. For example, for 3-jobs-3-machines JSP, a feasible chromosome can be encoded as [1 2 3 3 2 1 1 3 2].

2) Evaluation

To evaluate each chromosome, one needs to decode it to a solution. Gap extrusion method [12] is applied to produce movable scheme. The fitness can be calculated as:

$$f_j = 1/C_{ij} \tag{6}$$

j is any individual in the population.

3) Selection

Selection is the core of "survival of the fittest". In this step, those with higher fitness are selected to generate the offspring. Using roulette wheel selection, the probability of an individual to be selected is:

$$P_j = f_j \Big/ \sum_{j=1}^{N} f_j \tag{7}$$

N is the population size, f_j is the fitness of individual j.

4) Adaptive operator selection

4 frequently used crossover operators and 4 mutation operators are selected from literatures [11, 13~16] to form 16 combinations. Each operator combination is composed of one cross operator and one mutation operator. The adaptive operation selection includes two parts: (1) a credit assignment mechanism, which associates a reward with each operator combination and (2) a selection rule, which determines the combination to be used at each step.

a) Credit assignment

After each iteration, a certain credit is assigned to each combination of operators. It should assign a higher reward to those which have had a large positive impact on search process, while assign a lower credit (can be negative) to those which degrades the population fitness. Moreover, the recent exploration experience should be inherited fully, the sliding time window [17], therefore, is introduced in to the credit assessment.

Rewards are assigned as follows:

(1) Normalize the initial value of the fitness (f_0) as the first weight of all the combinations; and

(2) Set a time window, which is a two-dimensional list of W rows and H columns, H represents the number of combinations, W is the length of time window. When combination h is selected, it is applied to the current solution. The computed new fitness (f_h) is added to a FIFO list of size W, the FIFO means that when the W+1 fitness is appended into the window, the first added fitness will be removed from the window. All combinations share one list, the combination credit is updated to be the maximal fitness in the list. The expected reward for combination h is computed as follows:

$$Reward_h = max\{normalized(f_{ih}),\ i=1\cdots W,\ h=1\cdots H\} \tag{8}$$

b) Selection rule

The roulette wheel is employed to select the combination of operators. However, the roulette wheel depends on random number. As a result, it is possible to miss the current optimal combination. Therefore, we improve the roulette wheel selection process. In the odd iterations, the probability of combination h to be selected is:

$$P_h = Reward_h / \sum Reward_i,\ i=1,2,\cdots,H \tag{9}$$

Thus, we obtain that the member of best adaptation has the greatest probability to be selected.

In the even iteration, the best combination is selected with probability 1, to make sure that the best combination not to be missed.

B. The low-level design

1) Crossover operator

In this research, four crossovers are used for applying to hyper-heuristics: (1) C1: Precedence Operation Crossover (POX); (2) C2: Partial-Mapped Crossover (PMX); (3) C3: Partial Schedule Exchange Crossover (PSXX); (4) C4: Precedence Preservative Crossover (PPX).

2) Mutation operator

Four mutations are used here: (1) M1: Swap Mutation; (2) M2: Insertion Mutation; (3) M3: Displacement Mutation; (4) M4: Inversion Mutation.

3) The low-level heuristics

The low-level heuristics pool is structured with 16 combinations of 4 crossover operators and 4 mutation operators aforementioned (table I).

TABLE I. LOW-LEVEL HEURISTICS POOL

Low-level heuristics			
L1：C1/M1	L2：C1/M2	L3：C1/M3	L4：C1/M4
L5：C2/M1	L6：C2/M2	L7：C2/M3	L8：C2/M4
L9：C3/M1	L10：C3/M2	L11：C3/M3	L12：C3/M4
L13：C4/M1	L14：C4/M2	L15：C4/M3	L16：C4/M4

IV. EXPERIMENTS AND RESULTS

A. Design of experiments

Two experiments are carried out on a PC with Intel Core i5 processor, CPU frequency 2.6GHZ, 4GB memory, Windows7 operating system and coding with Matlab R 2012a.

The benchmark instance FT10 [18] is chosen as the date set. *In this experiment, population size is 500, generation number is 3000, crossover rate is 0.8 and mutation rate is 0.1. Different values of W are set. Each Algorithm runs 50 times.*

1) Experiment 1: the aim is to test the influnce of the size of the sliding window. Different size of time window are set: W=0,W=30,W=100,W=300.

2) Experiment 2: The aim is to make a comparis on with the traditional genetic algorithm to verify the effectiveness of the proposed hyper-heuristic algorithm.

B. Computational analysis

1) Experiment 1: Results are shown in table II. Column *A* represent the times of combination being selected as the best combination, column *B* represent the average makespan of the correspoding combination.

One can observe that the integration of the sliding time window shows better performance than no integration. Also, when time window is set, for example, W=30, the combinations whose cross operator is POX are more easily to be selected and the average completion time of these combinations are better than others. This suggests that POX is the best among the four crossovers. One can choose L1 or L2 as the most effective combination in solving FT10.

TABLE II. GENETIC BASED HYPER-HEURISTIC

LLH	W=0		W=300		W=100		W=30	
	A	*B*	*A*	*B*	*A*	*B*	*A*	*B*
L1	3	993	9	1005	8	999	11	993
L2	2	1000	6	1003	9	999	14	1001
L3	3	997	4	1001	8	1002	2	1012
L4	3	998	4	1003	8	1002	4	1003
L5	6	1005	5	1007	9	1007	3	998
L6	3	1004	4	1013	2	1001	7	1007
L7	4	1007	1	1007	0	0	3	1005
L8	6	1004	1	995	1	1021	4	998
L9	5	1021	0	0	1	1009	0	0
L10	4	1019	2	1025	1	1001	1	998
L11	6	1020	7	1016	2	1015	0	0
L12	5	1023	6	1017	1	1009	1	1010
L13	0	0	0	0	0	0	0	0
L14	0	0	1	990	0	0	0	0
L15	0	0	0	0	0	0	0	0
L16	0	0	0	0	0	0	0	0
Min.	-	974	-	967	-	977	-	982
Max.	-	1041	-	1037	-	1021	-	1019
Ave.	-	1010	-	1008	-	1003	-	1001
Var.	-	221	-	180	-	99	-	77

*2) Experiment 2:*Manually changing the crossover operator and mutation operator, traditional genetic algorithm is applied to FT10. Results are shown in table III.

TABLE III. TRADITIONAL GENETIC ALGORITHM

Crossover	Mutation	Min.	Max.	Ave.	Var.
C1	M1	967	985	978	25
	M2	958	982	968	71
	M3	973	1008	997	69
	M4	971	1007	991	66
C2	M1	985	1017	1003	71
	M2	985	1009	1001	57
	M3	990	1017	1005	58
	M4	996	1020	1007	44
C3	M1	1010	1043	1027	64
	M2	995	1041	1021	183
	M3	1016	1045	1029	44
	M4	1013	1208	1048	2944
C4	M1	949	1021	994	330
	M2	969	1059	1003	461
	M3	974	1026	998	192
	M4	969	1027	992	228

The results obtained by traditional genetic algorithm prove that all schemes with crossover POX (C1) are good. Among them, C1/ M2, is the best, C1/ M1, ranks the second, which proves that the conclusion obtained by the proposed hyper-heuristics is reasonable. When compared with some of the traditional genetic algorithms, the average makespan

obtained by hyper-heuristics (W=100 and W=30), performs more outstandingly. Even when W=0, the results are still better than that with the crossover C3, which proves that it is much better having many operators to select in GA than using a fixed operator. Also, human experts do not have to do much design work.

V. CONCLUSION

In this paper, we proposed a genetic based hyper-heuristic to solve JSP. The main contribution of this paper is a highly-automated approach to select genetic operators. The introduction of time window makes it more effective. The experiment result illustrates that the hyper-heuristic algorithm with the adaptive operation selection outperforms the traditional genetic algorithm whose genetic operators are prefixed.

REFERENCES

[1] P. Brucker, J. Bernd, and S, Bernd. "A branch and bound algorithm for the job-shop scheduling problem," Discrete applied mathematics, vol. 49, 1994, pp. 107-127.

[2] R. Tavakkoli-Moghaddam, F. Jolai, F. Vaziri, and P.K. Ahmed, A. Azaron, "A hybrid method for solving stochastic job shop scheduling problems," Applied Mathematics and Computation, vol. 170, 2005, pp. 185-206, doi:10.1016/j.amc.2004.11.036.

[3] B. Murovec and P. Šuhel, "A repairing technique for the local search of the job-shop problem," European Journal of Operational Research, vol. 153, 2004, pp. 220-238, doi:10.1016/S0377-2217(02)00733-6.

[4] K. Bülbül, "A hybrid shifting bottleneck-tabu search heuristic for the job shop total weighted tardiness problem," Computers & Operations Research, vol.38, 2011, pp. 967-983, doi:10.1016/j.cor.2010.09.015.

[5] R. Zhang, S.J. Song, C. Wu, "A hybrid artificial beecolony algorithm for the job shop scheduling problem," International Journal of Production Economics, vol. 141, 2013, pp. 167-178, doi: 10.1016/j.ijpe.2012.03.035.

[6] I. Essafi, Y. Mati and S. Dauzère-Pérès, "A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem," Computers & Operations Research, vol. 35, 2008, pp. 2599-2616, doi:10.1016/j.cor.2006.12.019.

[7] M. Watanabe, K. Ida and M. Gen, "A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem," Computers & Industrial Engineering, vol. 48, 2005, pp. 743-752, doi:10.1016/j.cie.2004.12.008.

[8] E.K. Burke, M. Gendreau, M. Hyde, et al., "Hyper-heuristics: a survey of the state of the art," Journal of the Operational Research Society, vol. 64, 2013, pp. 1695–1724, doi:10.1057/jors.2013.71.

[9] J.A. Soria-Alcaraz, G. Ochoa, J. Swan, M. Carpio, H. Puga and E.K. Burke, "Effective learning hyper-heuristics for the course timetabling problem," European Journal of Operational Research, vol. 238, 2014, pp. 77-86, doi:10.1016/j.ejor.2014.03.046.

[10] G. Koulinas, L. Kotsikas and K. Anagnostopoulos, "A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem," Information Sciences, 2014, doi:10.1016/j.ins.2014.02.155.

[11] M. Gen, Y. Tsujimura and E. Kubota, "Solving job-shop scheduling problems by genetic algorithm," Systems, Man, and Cybernetics, 1994. Humans, Information and Technology,1994 IEEE International Conference on. IEEE, 1994, pp. 1577-1582.

[12] X.L. Wu, S.M. Wu, "An elitist quantum-inspired evolutionary algorithm for the flexible job-shop scheduling problem," Journal of Intelligent Manufacturing. March. 2015, doi:10.1007/s10845-015-1060-6.

[13] C.Y. Zhang, P.G. Li, Y.Q. Rao and S.X. Li, "A new hybrid GA/SA algorithm for the job shop scheduling problem," Evolutionary Computation in Combinatorial Optimization. Springer Berlin Heidelberg, 2005, pp. 246-259, doi:10.1007/978-3-540-31996-2_23.

[14] D.E. Goldberg, R. Lingle, "Alleles, loci and the traveling salesman problem," Proceedings of the first international conference on genetic algorithms and their applications. Lawrence Erlbaum Associates, Publishers, 1985, pp. 154-159.

[15] C. Bierwirth, D.C. Mattfeld and H. Kopfer, "On permutation representations for scheduling problems," Parallel Problem Solving from Nature—PPSN IV. Springer Berlin Heidelberg, 1996, pp. 310-318, doi:10.1007/3-540-61723-X_995.

[16] B. Akay and X. Yao. "Recent Advances in Evolutionary Algorithms for Job Shop Scheduling," Automated Scheduling and Planning, Springer Berlin Heidelberg, 2013, pp. 191-224, doi: 10.1007/978-3-642-39304-4_8.

[17] Á. Fialho, L.D. Costa, M. Schoenauer and M. Sebag. "Analyzing bandit-based adaptive operator selection mechanisms," Annals of Mathematics and Artificial Intelligence, vol. 60, 2010, pp. 25-64,doi: 10.1007/s10472-010-9213-y.

[18] H. Fisher and G.L. Thompson. "Probabilistic learning combinations of local job-shop scheduling rules," Industrial Scheduling, 1963, pp. 225-251.