The study of comparisons of three crossover operators in genetic algorithm for solving single machine scheduling problem

Quan OuYang, Hongyun XU [a]*

School of Mathematics & Computer Science, JiangHan University, China

[a]xhy1978@163.com

Keywords: crossover operator, single machine scheduling, genetic algorithm

Abstract. genetic algorithm is a common method for solving combinatorial optimization problems and the selection of crossover operators in genetic algorithm will directly affect the performance of the algorithm. In this paper, we compare the performance of three crossover operators, partially mapped crossover operator (PMX), order based crossover operator (OBX), and adaptation of the edge recombination crossover operator (aERX), under same genetic algorithm framework in solving the un-weighted single machine scheduling problem with sequence dependent setup times. It is concluded that the performance of PMX crossover operator is better than the other two crossover operators from the computational results.

Introduction

Genetic algorithm is an efficient heuristic algorithm, which can be used to solve the problem by imitating biological evolution. In genetic algorithm the crossover operator is the main method to generate new individuals, and the choice of crossover operator directly affects the efficiency of genetic algorithm. Scheduling problem is a large class of problems and there are many scheduling problems have been proved to be NP hard. For example, the un-weighted single machine scheduling problem without setup times has proved to be NP-hard in [1,2]. In this paper, under the same framework of genetic algorithm three different crossover operators are used to solve un-weighted single machine scheduling problem with sequence-dependent setup times and the more suitable crossover operator is selected by comparing the results in solving single machine scheduling problem.

Single machine scheduling problem description

Single machine scheduling problem solved in this paper can be simply described as: N independent jobs need to be processed in one machine and every job has its expected due date and processing time and other parameters. In addition, the different single machine scheduling problems have different constraints and the objective of which is to find a job sequence which makes the total cost is minimal. In this paper the un-weighted single machine scheduling problem with sequence-dependent setup times expressed as $1|s_{ij}|T_j$ is solved. The objective of $1|s_{ij}|T_j$ is to find a job sequence which minimizes the total tardiness T.

Genetic algorithm

Genetic algorithm is generated by imitating biological evolution, which has strong robustness and was adapted to the wide range. The main implementation process of genetic algorithm includes the design and selection of coding, population initialization, selection strategy, crossover operator and mutation. In this paper the initial population is generated randomly and the benefit of this is that it can produce a population with good diversity. The roulette strategy is adopted for choosing parents from the population as the selection strategy.

Crossover operator

In genetic algorithm, the crossover operator is used to generate the new offspring and the design of the crossover operator must satisfy the rule of feasibility, property inheritance, completeness and nonredundancy. In general, a good crossover operator generates the offspring not only have large differences between parents and them but also inherit the sufficient good genes from their parents. Through the study of some crossover operators, we compare the performance of three crossover operators, partially mapped crossover operator (PMX), order based crossover operator (OBX), and edge recombination crossover operator (aERX), in solving the problem of single machine scheduling problem.

PMX. Suppose P1 and P2 are two parent individuals and C is one child individual produced by the crossover operator introduced in this paper. Partially mapped crossover operator (PMX) was proposed by Goldberg and Lingle [3] in 1985, the procedures of which are as follows:

Step 1: randomly select two cut points on P1 and P2;

Step 2: construct a mapping table T1 for the jobs between two cut points in P1 and P2;

Step 3: copy the jobs between two cut points in P2 to C by keeping job's position in P2;

Step 4: copy the jobs except between two cut points in P1 to C by keeping jobs' position in P1;

Step 5: keep the jobs between two cut points unchanged, if there are same jobs to which and replaced the repetitive jobs according to the mapping table T1 until there is no repetition jobs in C.

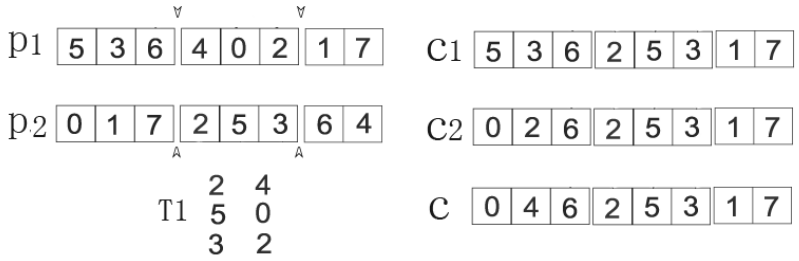

Fig. 1. Partially mapped crossover operator (PMX)

As shown in Fig. 1, P1 and P2 are two parent individuals, P1={5,3,6,4,0,2,1,7} and P2={0,1,7,2,5,3,6,4}. First, we choose two cut points in P1 and P2 randomly: one is between positions 3 and 4, the other is between positions 6 and 7. The jobs between two points in P1 and P2 are {4, 0, 2} and {2, 5, 3} and the mapping table T1 is constituted by the one to one mapping, {2,4}, {5,0}, {3,2}, between {4, 0, 2} and {2, 5, 3}. The jobs between two cut points in P2 are {2, 5, 3} which are copied to C by keeping their job positions in P2 and the jobs except between two cut points in P1 are {5, 3, 6, 1, 7} which are copied to C by keeping their job positions in P1 too, and we got C={5,3,6,2,5,3,1,7}. After that, we observed that there are double 5 and 3 in C. Then, remain the jobs in two cut points {2, 5, 3} unchanged and replaced the other 5 and 3 by 0 and 2 according the mapping table T1, and C={0, 2, 6, 2 , 5, 3 ,1, 7}. Keep checking if there are same jobs to the jobs between two cut points {2, 5, 3}. We can find there are double 2 in C and keep {2, 5, 3 } unchanged and replaced the other 2 by 4 according to the mapping table T1. Finally, C generated by PMX is {0, 4, 6, 2, 5, 3, 1, 7}.

OBX. Order based crossover operator (OBX) was proposed by Syswerda [4] which is a slight variation of the PBX operator. The OBX's working procedures are as follows:

Step 1: randomly select a set of positions in P1;

Step 2: copy P2's jobs except these in selected positions in P1 to C by keeping their positions in P2.

Step 3: copy the jobs in selected positions of P1 to C by preserving jobs' order in P1.

Fig. 2 illustrates how the OBX operator works. In this example, the selected positions are 1, 3, 6, and 8. Thus, the selected jobs in P1 and are {5, 6, 4, 7} and except {5, 6, 4, 7} the remaining jobs in P2 are {0, 1, 2, 3}. Then, we copy {0, 1, 2, 3} to the corresponding positions to C by keeping their positions in P2. Next, we place {5, 6, 4, 7} to C according to the jobs' order in P1. Thus, the constructed offspring are C={0, 1, 5, 2, 6, 3, 4, 7} .

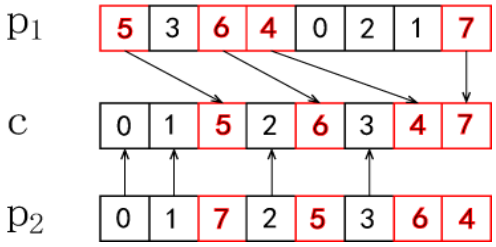

Fig. 2. Order based crossover operator

aERX. Adaptation of the edge recombination crossover operator (aERX) was developed for the travelling salesman problem and presented in detail in [5]. The working procedures of aERX are as shown below:

Step 1: a successor list is constructed according to the sequential order of jobs in P1 and P2, the example of a successor list is shown in Fig. 3;

Step 2: randomly select a job as the first job of C;

Step 3: find two successor jobs of the current job in the successor list;

Step 4: randomly choose one of the two jobs with probability 0.5 as a new member of C;

Step 5: Repeat steps 3 and 4 until all jobs have been chosen without repetition in C.

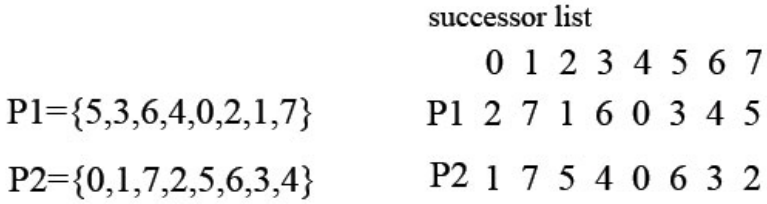

Fig. 3. Successor list

Experimental scheme and result analysis

In order to compare the performance of the three different crossover operators, PMX, OBX, and aERX, in solving the $1|s_{ij}|\sum T_j$ problem, we produce three genetic algorithms by using the same initial population generation method, selection strategy and mutation with three different crossover operators introduced above, PMX, OBX, and aERX, and apply the three genetic algorithms to solve the 32 instances generated by Rubin and Ragatz [6] of the $1|s_{ij}|\sum T_j$ problem. The 32 instances include 4 groups with 15, 25, 35, and 45 jobs respectively and every group has 8 instances. We programmed the genetic algorithms in C++ and compiled it on a PC running Linux with 2.6 GHz CPU and 2.0 Gb RAM. We solved the 32 instances independently for 50 times using different random seeds subject to a time limit of 30 CPU seconds for the three genetic algorithms.

We summarize the comparisons of the three genetic algorithms' best results with the optimal solutions (OPT) for the 32 public instances in Table 1, in which the numbers of equal and worse solutions compared with the OPT are presented. In Table 1, comparing the results of the three genetic algorithms (PMX, OBX, and aERX) with OPT, the numbers of equal solutions are 15, 16, and 12 respectively, and the number of worse solutions are 17, 16, and 20. We can find that in columns OBX and PMX the numbers of equal solutions 16 and 15 are better than column aERX.

Table 1 Summary of comparing the three GAs' best results with optimal solutions

	PMX	**OBX**	**aERX**
Number of equal solutions	15	16	12
Number of inferior solutions	17	16	20
Number of all solutions	32	32	32

However, it is not enough to demonstrate which crossover operator is the best to solve the $1|s_{ij}|\sum T_j$ problem in genetic algorithm only from the number of equal solutions. In order to gain a deeper understanding of the performance of the three crossover operators, we summarize in Table 2 the average differences of the three genetic algorithms' best results with respect to the OPT in each group of the instances(i.e., Δ aERX = (aERX-OPT)/OPT×100%). In Table 2, columns 2-4 give the average differences of the best results of the three genetic algorithms relative to OPT respectively.

Table 2 Average differences of the three GAs' best results with respect to the OPT in each group

Job number	**Δ PMX**	**Δ OBX**	**Δ aERX**
15	0.56%	0.00%	2.68%
25	2.31%	5.01%	10.67%
35	85.31%	187.13 %	220.13%
45	40.47%	115.09 %	99.75%
	32.16%	76.81%	83.31%

From Table 2, we observe that the average differences of column PMX is smaller than the other two genetic algorithms except when the job number is 15. Last row presents the average differences for all the instances. From last row, we can find that the total average difference for columns aERX, OBX, and PMX are 32.16%, 76.81%, and 83.31% respectively and the column PMX is better than the other two columns.

Conclusion

In genetic algorithm, the appropriate choice of the crossover operator will directly affect the ability of the algorithm to search the best solution. In this paper, we present a systematic comparison of three genetic algorithms with three different crossover operators (PMX, OBX, and aERX) for solving the $1|s_{ij}|\sum T_j$ problem. From the experiment results, we find that the genetic algorithm with crossover operator PMX can achieve competitive performance than the other two crossover operators OBX and aERX.

Acknowledgments

The research was supported by Hubei Provincial Department of Education under grant number B2014077

References

[1] Lawler EL.A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. Annals of discrete Mathematics, 1977, 1:331–342.

[2]Du J, Leung J Y T, Mathematics of operations research, 1990, 15(3):483–495.

[3] D.E. Goldberg, R. Lingle, Alleles, loci, and the traveling salesman problem, in: J.J. Grefenstette (Ed.), Proceedings of the First International Conference on Genetic Algorithms and their Applications, Lowrence Erlbaum, Hillsdale, 1985, pp. 154– 159.

[4] G. Syswerda, Schedule Optimization Using Genetic Algorithms, in: L. Davis (Ed.), Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1990.

[5] Vinicius A. Armentano & Renata Mazzini (2000): A genetic algorithm for scheduling on a single machine with set-up times and due dates, Production Planning & Control: The Management of Operations, 11:7, 713-720

[6] Rubin P A, Ragatz G L, Computers & Operations Research, 1995, 22(1):85–99