Performance evaluation of different types of particle representation procedures of Particle Swarm Optimization in Job-shop Scheduling Problems

2016 IOP Conf. Ser.: Mater. Sci. Eng. 114 012055

(http://iopscience.iop.org/1757-899X/114/1/012055)

IOP Conf. Series: Materials Science and Engineering **114** (2016) 012055 doi:10.1088/1757-899X/114/1/012055

Performance evaluation of different types of particle representation procedures of Particle Swarm Optimization in Job-shop Scheduling Problems

Nurul Izah Anuar[1] **and Adi Saptari**[2]
[1]Centre for Diploma Programmes, Multimedia University, Jalan Ayer Keroh Lama, 75450 Ayer Keroh, Melaka, Malaysia
[2]Faculty of Manufacturing Engineering, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia

E-mail: nurulizah.anuar@mmu.edu.my

Abstract. This paper addresses the types of particle representation (encoding) procedures in a population-based stochastic optimization technique in solving scheduling problems known in the job-shop manufacturing environment. It intends to evaluate and compare the performance of different particle representation procedures in Particle Swarm Optimization (PSO) in the case of solving Job-shop Scheduling Problems (JSP). Particle representation procedures refer to the mapping between the particle position in PSO and the scheduling solution in JSP. It is an important step to be carried out so that each particle in PSO can represent a schedule in JSP. Three procedures such as Operation and Particle Position Sequence (OPPS), random keys representation and random-key encoding scheme are used in this study. These procedures have been tested on FT06 and FT10 benchmark problems available in the OR-Library, where the objective function is to minimize the makespan by the use of MATLAB software. Based on the experimental results, it is discovered that OPPS gives the best performance in solving both benchmark problems. The contribution of this paper is the fact that it demonstrates to the practitioners involved in complex scheduling problems that different particle representation procedures can have significant effects on the performance of PSO in solving JSP.

1. Introduction

The Job-shop Scheduling Problem (JSP) basically entails a collection of machines occupied by a set of jobs with a predefined machine order. An operation represents the processing of a job on a certain machine. A delegation of operations of a job to time intervals on a machine is called a schedule. A good or optimal schedule is able to identify the best order of operations scheduled on all machines in order to optimize certain performance measures [1].

Particle Swarm Optimization (PSO) is one of the evolutionary computation techniques developed by Kennedy and Eberhart [2]. It is a population-based search algorithm and is initialized with a population of random solutions called particles [3]. The original algorithm was discovered through simplified social model simulation [4]. Each particle in PSO is associated with a velocity. The particle flies through the search space with a velocity which is dynamically adjusted according to its own flying experience and its companions' flying experiences. Hence, the particles have a tendency to fly towards the better search area over the course of searching process.

Before we can apply the PSO to solve the JSP, one common and important issue to be addressed is how we can represent each particle of PSO as a schedule in JSP [5]. A particle in PSO represents one potential solution i.e. the particle represents one schedule that may potentially be the solution that we are looking for. PSO is mainly applied to solve continuous optimization problems whereas JSP is a discrete permutation problem. This is where particle representation procedures comes into play. Particle representation procedures refer to the mapping between the particle position in PSO and the scheduling solution in JSP. It is an important step to be carried out so that each particle in PSO can represent a schedule in JSP. Besides, JSP contains precedence constraints that must be satisfied in order for the schedule to be feasible. Otherwise, a suitable repair method is required in order to convert the infeasible schedule to a feasible one [6]. The particle representation procedures that are chosen in this study will produce feasible schedules only and thus, do not involve any repair method.

In this paper, three different types of particle representation procedures such as Operation and Particle Position Sequence (OPPS), random keys representation and random-key encoding scheme have been used for determining the minimum makespan of FT03 and FT10 problems. Afterwards, a comparative study of these procedures are also carried out in order to identify the procedure that produce the best performance. The paper is organized in the following manner: first, the particle representation procedures are described and the problem description is presented. Finally, the results, discussion and conclusions are provided.

2. Particle representation procedures

A particle representation procedure is employed to establish the mapping between the particle position in PSO and the scheduling solution in JSP. The key issue is to find a suitable method in order for each particle in PSO to be able to represent a schedule in JSP of a specific problem. In this paper, three procedures are selected to be evaluated and compared. The three procedures are explained in detail below:

2.1. Operation and Particle Position Sequence (OPPS)

This method to represent the particle as a schedule was proposed by Liu [7]. This technique starts with the particle's position in random continuous numbers being tied with the position sequence (in ascending order). If we sort the particle's position in an ascending or descending order, the position sequence would also change accordingly. Each position in this new position sequence will then be assigned on each machine in turn in accordance to the precedence constraints of JSP.

2.2. Random Keys Representation

This technique was introduced by Pongchairerks and Kachitvichyanukul [8]. This technique starts with sorting the particle's position in ascending order. This new order of particle's position then is tied with the job sequence. Now, the particle's position is sequenced back in the original order, where the job sequence would also change accordingly. Each job in this new job sequence will then be assigned on each machine in turn in accordance to the precedence constraints of JSP.

2.3. Random-Key Encoding Scheme

This method was introduced by Lin, Horng, Kao, Chen, Run, Chen, Lai and Kuo in 2010 [9]. This technique starts with sorting the particle's position in ascending order. This new order of particle's position then is tied with the position sequence. Now, the particle's position is sequenced back in the original order, where the position sequence would also change accordingly. Each position in this new position sequence will then be assigned on each machine in turn in accordance to the precedence constraints of JSP.

3. Problem description

In order to test the procedures described in the previous section, two benchmark problems are used which are taken from the OR-Library website [10]. The first problem consists of 6 machines and 6

jobs with 6 operations in each machine. The details of the problem called FT06 is described in table 1. In the table, each row is a job sequence with processing time in parentheses.

Table 1. FT06 scheduling problem.

Job no.	Sequence of machines (processing time)					
1	2 (1)	0 (3)	1 (6)	3 (7)	5 (3)	4 (6)
2	1 (8)	2 (5)	4 (10)	5 (10)	0 (10)	3 (4)
3	2 (5)	3 (4)	5 (8)	0 (9)	1 (1)	4 (7)
4	1 (5)	0 (5)	2 (5)	3 (3)	4 (8)	5 (9)
5	2 (9)	1 (3)	4 (5)	5 (4)	0 (3)	3 (1)
6	1 (3)	3 (3)	5 (9)	0 (10)	4 (4)	2 (1)

As could been seen from table 1, there are 6 jobs which need to be scheduled using 6 machines. There are 6 rows (which represent 6 jobs with processing time in parentheses) and 6 columns (which correspond to 6 machines). Each row is a permutation of numbers representing the sequence of machines that a job must visit. For example, Job 1 must initially be processed on Machine 2 for 1 unit of processing time; next it must go to Machine 0 for 3 units of time; and later requires Machine 1 for 6 time units and so on (the machines are numbered starting from 0).

The second problem consists of 10 machines and 10 jobs with 10 operations in each machine. The details of the problem called FT10 is described in table 2. There are ten rows and ten columns representing ten jobs and ten machines, respectively. For example, Job 3 must go to Machine 1 first for 91 time units, and then it must be processed on Machine 0 for 85 time units, then go to Machine 3 for 39 time units and so on.

Table 2. FT10 scheduling problem.

Job no.	Sequence of machines (processing time)									
1	0 (29)	1 (78)	2 (9)	3 (36)	4 (49)	5 (11)	6 (62)	7 (56)	8 (44)	9 (21)
2	0 (43)	2 (90)	4 (75)	9 (11)	3 (69)	1 (28)	6 (46)	5 (46)	7 (72)	8 (30)
3	1 (91)	0 (85)	3 (39)	2 (74)	8 (90)	5 (10)	7 (12)	6 (89)	9 (45)	4 (33)
4	1 (81)	2 (95)	0 (71)	4 (99)	6 (9)	8 (52)	7 (85)	3 (98)	9 (22)	5 (43)
5	2 (14)	0 (6)	1 (22)	5 (61)	3 (26)	4 (69)	8 (21)	7 (49)	9 (72)	6 (53)
6	2 (84)	1 (2)	5 (52)	3 (95)	8 (48)	9 (72)	0 (47)	6 (65)	4 (6)	7 (25)
7	1 (46)	0 (37)	3 (61)	2 (13)	6 (32)	5 (21)	9 (32)	8 (89)	7 (30)	4 (55)
8	2 (31)	0 (86)	1 (46)	5 (74)	4 (32)	6 (88)	8 (19)	9 (48)	7 (36)	3 (79)
9	0 (76)	1 (69)	3 (76)	5 (51)	2 (85)	9 (11)	6 (40)	7 (89)	4 (26)	8 (74)
10	1 (85)	0 (13)	2 (61)	6 (7)	8 (64)	9 (76)	5 (47)	3 (52)	4 (90)	7 (45)

The assumptions considered in this paper are:

- Setup times of machines are negligible.
- Machines are independent of each other.
- Jobs are independent of each other.
- A machine cannot process more than one job at a time. No jobs may be processed on more than one machine at a time.
- There are no precedence constraints among the operations of different jobs.

4. Results and discussion

In the experiments, PSO for JSP is programmed using MATLAB and run on a 3.1GHz Intel Core PC with 4GB of RAM running Windows 8. The parameters used during the experimental process are as follows: The population size is set to 40 and 50 for FT06 and FT10 problems, respectively. The inertia weight is decreased linearly from 0.9 at the beginning of the run and 0.4 at the end of the run. Both

acceleration constants are set to 2.0. The maximum of iterative generations is set to 1000 and each instance is randomly performed 10 times.

The computational results for OPPS, random keys representation and random-key encoding scheme are shown in table 3 to be compared. The solution quality is measured in terms of the minimum makespan of PSO and the percent relative increase in minimum makespan with respect to the optimal solutions. In table 3, best solution denotes the minimum makespan and mean deviation denotes the average percent relative increase in minimum makespan with respect to the optimal solution after 10 runs.

The optimal solution to be compared for FT06 problem is 55, which is obtained from the paper by Klemmt, Horn, Weigert, and Wolter [11] who used exact methods to solve the problem. The best solution obtained by all 3 procedures is also 55; hence, all 3 procedures managed to obtain the optimal solution. However, in terms of mean deviation, OPPS performs the best out of all 3 procedures with a mean deviation of 0.55%; in other words, out of 10 runs, there is only 1 run where the solution is not equivalent to the optimal solution. Meanwhile, random keys representation performs better than random-key encoding scheme in terms of mean deviation with 2.91%.

The optimal solution to be compared for FT10 problem is 930, which is obtained from the paper by Carlier and Pinson [12] who used exact methods to solve the problem. The best solution obtained by OPPS is 1029 which outperforms the other 2 procedures; hence, OPPS is able to obtain a near-optimal solution with 10.65% deviation from the optimal solution. In terms of mean deviation, OPPS performs the best out of all 3 procedures with a mean deviation of 18.90%. Meanwhile, random keys representation performs better than random-key encoding scheme in terms of best solution and mean deviation with 1089 and 23.53%, respectively.

Table 3. Computational results for OPPS, random keys representation and random-key encoding scheme.

Problem instance	Optimal solution	OPPS		Random keys representation		Random-key encoding scheme	
		Best solution	Mean deviation (%)	Best solution	Mean deviation (%)	Best solution	Mean deviation (%)
FT06	55	55	0.55	55	2.91	55	3.45
FT10	930	1029	18.90	1089	23.53	1119	28.42

5. Conclusion

In this paper, 3 different types of particle representation procedures in PSO in the case of solving JSP are studied. The performance of OPPS, random keys representation and random-key encoding scheme are tested on FT06 and FT10 benchmark problems available in the OR-Library. The performance measure considered in this study is the minimum makespan by the use of MATLAB software. Based on the experimental results, it is discovered that OPPS gives the best performance in solving both benchmark problems, as compared to the random keys representation and random-key encoding scheme.

Through the evaluation and comparison of these 3 procedures, the authors manage to gain some insight into different particle representation procedures in terms of how it can significantly affect the performance of PSO in solving JSP. The procedures are shown to produce considerably different level of performances for similar problem instances, where OPPS manage to surpass the other 2 procedures with a substantial improvement in achieving the optimal solutions of 2 benchmark problems. Thus, the particle representation procedure should be considered as one of the important parameters when working with PSO in solving JSP.

Future works will involve analyzing the mechanism of each procedure in order to investigate what makes one procedure better than the other. Besides, other particle representation procedures will also

be tested in order to find out which one can yield a competitive performance. Since only 2 benchmark problems are tested, more benchmark problems in the OR library will be used to evaluate the procedures in order to verify whether the performance depends on the problems being solved.

References

[1] Pezzella F, Morganti G and Ciaschetti G 2008 A genetic algorithm for the Flexible Job-shop Scheduling Problem *Comput. Oper. Res.* **35** 3202–12
[2] Kennedy J and Eberhart R 1995 Particle swarm optimization *Neural Networks, 1995 Proceedings, IEEE Int Conf* vol 4 (Perth: IEEE) pp 1942–8
[3] Eberhart R C and Shi Y 2001 Particle swarm optimization: developments, applications and resources *Proc 2001 Congr Evol Comput* vol 1 (Seoul: IEEE) pp 81–6
[4] Błażewicz J, Domschke W and Pesch E 1996 The job shop scheduling problem: conventional and new solution techniques *Eur. J. Oper. Res.* **93** 1–33
[5] Xia W and Wu Z 2006 A hybrid particle swarm optimization approach for the job-shop scheduling problem *Int. J. Adv. Manuf. Technol.* **29** 360–6
[6] Yen G G and Ivers B 2009 Job shop scheduling optimization through multiple independent particle swarms *Int. J. Intell. Comput. Cybern.* **2** 5–33
[7] Liu Z 2007 Investigation of Particle Swarm Optimization for Job Shop Scheduling Problem *3rd Int. Conf. Nat. Comput. (ICNC 2007)* vol 3 (Haikou: IEEE) pp 799–803
[8] Pongchairerks P and Kachitvichyanukul V 2009 A Particle Swarm Optimization algorithm on Job-Shop Scheduling Problems with multi-purpose machines *Asia-Pacific J. Oper. Res.* **26** 161–84
[9] Lin T-L, Horng S-J, Kao T-W, Chen Y-H, Run R-S, Chen R-J, Lai J-L and Kuo I-H 2010 An efficient job-shop scheduling algorithm based on particle swarm optimization☆ *Expert Syst. Appl.* **37** 2629–36
[10] Beasley J E 1990 OR-Library http://people.brunel.ac.uk/~mastjjb/jeb/info.html (accessed January 1, 2015)
[11] Klemmt A, Horn S, Weigert G and Wolter K-J 2009 Simulation-based optimization vs. mathematical programming: a hybrid approach for optimizing scheduling problems *Robot Comput. Integr. Manuf.* **25** 917–25
[12] Carlier J and Pinson E 1989 An algorithm for solving the job-shop problem *Manage. Sci.* **35** 164–76