ISSN: 2442-6571

# An evolutionary approach for solving the job shop scheduling problem in a service industry

Milad Yousefi[a,1], Moslem Yousefi[b,2,*], Danial Hooshyar[c,3], Jefferson Ataide de Souza Oliveira[a,4]

[a]*Departamento de Engenharia Mecânica, Universidade Federal de Minas Gerais - UFMG, Brazil*
[b] *Centre of Advanced Mechatronics and Robotics, College of Engineering, University Tenaga Nasional (UNITEN),*
[c] *Department of Software Engineering, Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia*
[1] *Milad@ufmg.br;* [2] *Moslem_yousefi@yahoo.com*;* [3] *Danial.hooshyar@gmail.com;* [4] *jd12120@my.bristol.ac.uk*

ARTICLE INFO

*Article history:*
Received March 9, 2015
Revised March 23, 2015
Accepted March 29, 2015

*Keywords:*
Scheduling
Job shop scheduling problem
Optimization
Discrete particle swarm optimization

ABSTRACT

In this paper, an evolutionary-based approach based on the discrete particle swarm optimization (DPSO) algorithm is developed for finding the optimum schedule of a registration problem in a university. Minimizing the makespan, which is the total length of the schedule, in a real-world case study is considered as the target function. In order to clarify the problem and the proposed solution a small instance discusses then the problem with the real data is solved. Since the selected case study has the characteristics of job shop scheduling problem (JSSP), it is categorized as a NP-hard problem which makes it difficult to be solved by conventional mathematical approaches in relatively short computation time.

## I. Introduction

Scheduling problems in both manufacture and service fields have many industrial applications therefore, they received many attraction in recent decades. There are different types of scheduling problems such as single machine scheduling, open shop scheduling and flow shop scheduling which have proved to be NP-hard when combined them with different constraints and objectives [1]. The job shop scheduling problem (JSSP) because of its nature is considered to be the most difficult problem in this field and probably the most studied among scheduling problems [2].

The first attempts to solve JSSP appeared in the last few decades of the 20th century. The exact methods like integer programming [3], dynamic programming [4] are presented during this period. Hence, these methods are limited by size of problems and problems with more than 10 jobs and 10 machines exact methods take large amount of computational times [5]. Afterwards, the heuristic methods such as the branch and bound method [6]. Generally, approximation methods can achieve near optimal solutions therefore, they are not guarantee to attain optimal solutions [7][8]. Meeran and Morshed [9] categorized approximation techniques which are used to find JSSP solution in four categories: priority dispatch rules [10]–[12], bottleneck based heuristics [13]–[16], artificial intelligence and local search. However, some of these approximate techniques also need huge amount of computation time to reach their best solution. Most of scheduling problems are concentrated on machines therefore, this study tries to apply job shop scheduling problem on an uncommon problem that is in service level instead of machine level.

The rest of this paper is prepared as follows. In section 2, the selected case study is presented. The proposed method that includes the solution representation and the proposed discrete particle swarm optimization (DPSO) algorithm is presented in section 3. The computational results are presented in section 4. Finally, the conclusions are reported in Section 5.

## II. Case study

This study tries to find the near optima solution for a scheduling in a real world case study. In this case, new students of a university need to meet different operators in different sections for example, medical check, and documents check, insurance and etc. Traditionally universities use first come first serve in their registration sections, without considering other section therefore, students will face long queues in a section while there are only few students in other sections. The main objective of this study is to have a comprehensive look at the registration process and assign students in a way that all students pass all registrations section in a shortest time. Each student may meet same operator for different reason and each operator can only serve one and only one student at the time. Total number of students are 167 therefore, in order to explain the problem and the proposed solution an example of 4 students will be presented in this part. Table 1 shows the operators sequence matrix and table 2 demonstrate the operation time for each operation.

Table 1. The sequence of operators

| | **Operations** | | | |
|---|---|---|---|---|
| **Student** | ***1*** | ***2*** | ***3*** | ***4*** |
| S1 | $O_1$ | $O_2$ | $O_3$ | $O_4$ |
| S2 | $O_3$ | $O_1$ | $O_4$ | $O_2$ |
| S3 | $O_1$ | $O_3$ | $O_2$ | $O_4$ |
| S4 | $O_3$ | $O_2$ | $O_4$ | $O_1$ |

Table 1 shows in this example there are 4 students and each of them needs to be processed in 4 sections. In table 1, O stand for operator (section) subsequently, $O_1$ stand for operator 1. S is stand for Student.

Table 2. Processing time for each student in each section.

| | **Operations** | | | |
|---|---|---|---|---|
| **Student** | ***1*** | ***2*** | ***3*** | ***4*** |
| S1 | 3 | 3 | 2 | 2 |
| S2 | 1 | 5 | 3 | 2 |
| S3 | 3 | 2 | 3 | 4 |
| S4 | 2 | 4 | 3 | 3 |

Table 2 shows the processing time for each student in each section. Table 3 each student for each operation should meet which operator.

Table 3. Students sequence for a 4 * 4 problem.

| | **Operations** | | | |
|---|---|---|---|---|
| **Operators** | ***1*** | ***2*** | ***3*** | ***4*** |
| O1 | S3 | S1 | S2 | S4 |
| O2 | S4 | S3 | S1 | S2 |
| O3 | S4 | S3 | S2 | S1 |
| O4 | S4 | S3 | S2 | S1 |

The objective of this study is minimizing the makespan of the schedule. The makespan of is calculated by Eq (1):

$$C_{Max} = min\,(\text{Cmax}) = min_{feasibleschedules}\,(\max\,(t_{ik} + \tau_{ik}\,) \qquad (1)$$

$$(\forall Si \in S, S_k \in O)$$

## III. The proposed method

The problem at hand is a discrete NP-hard problem [17], [18]. To provide a robust EA for solving this problem, a discrete particle swarm optimization approach is implemented. This variant is based on discrete PSO proposed by Kennedy and Eberhart [19]. Similar to other population-based EAs, a specific number of possible solutions, referred to as particles in PSO terminology, are generated stochastically and then evaluated based on the objective function. In particle swarm optimization, each potential solution is a particle. Particles fly through the apace of the problem by following the particle that has the optimum solution currently. The original PSO is based on social behavior of bird flocking.

In PSO, each particle has coordinates in the problem space. These coordinates are associated with the best solution that achieved until now. The particle keeps fly on its way. The best value that achieve by particle is stored. This solution that particle keeps track of it called personal best "*pbest*". The other value that is tracked by particles is "*lbest*" that is the best solution (value) that is achieved till now by any of particles. The other "best" in PSO is global best that is called "*gbest*". Global best is a particle that takes all the population as its topological neighbors.

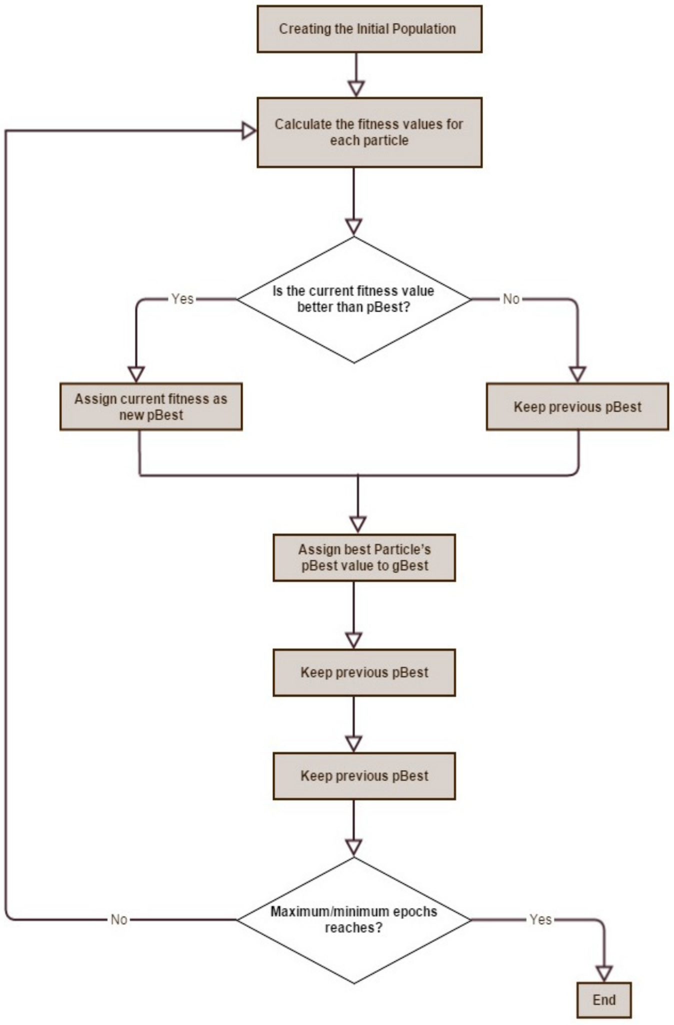
Creating the Initial Population
Calculate the fitness values for each particle
Is the current fitness value better than pBest?
Yes
No
Assign current fitness as new pBest
Keep previous pBest
Assign best Particle's pBest value to gBest
Keep previous pBest
Keep previous pBest
Maximum/minimum epochs reaches?
No
Yes
End

Fig. 1. The flowchart of particle swarm optimization

In particle swarm optimization in order to achieve the best solution, in each step all particles' location is changed toward its personal best and local best. Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *lbest* locations. Figure 1 shows the how the algorithm works.

From the above schedule a particle such as [1 3 1 4 2 3 2 3 1 4 1 3 4 2 2 4] could be a possible solution for the instance problem. This particle shows that the first operation of the first student should be done at first after that the first operation of third student should be done. Table 1 shows each operation for registration of each student should be done by which operator (section). There are 4 operators in this case for example first operation of first student should be done by operator number 3.

## IV. Numerical results

Similar to other EAs, PSO starts with creation of its initial population. In order to choose a proper population size, the algorithm is executed with different population sizes and the corresponding near-optimum solutions are calculated. The algorithm is run with 20, 50, 100, 150 and 200 numbers of particles as the population. The presented result of the case is the best result achieved in 10 executions of the algorithm. As it is demonstrated in Fig. 1, the attained result improves when the population size increases. Although, the changes are radical when the population is very small, they become negligible in large population sizes. In this study, since the variation of the achieved near-optimum solution is not considerable after increasing the population size more than 100 and to avoid additional computational costs, the population size is set to 100 particles.

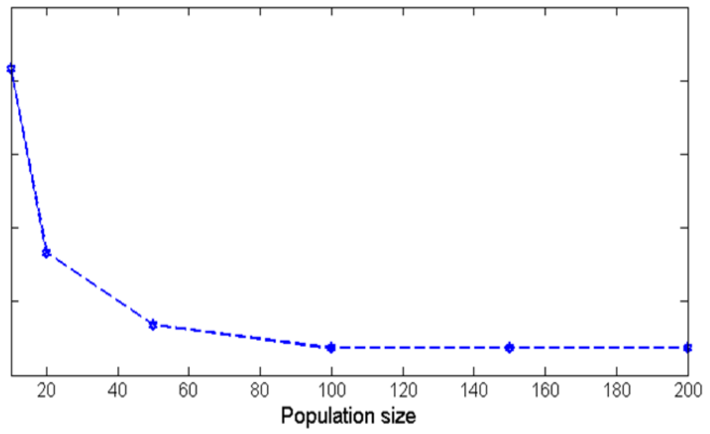

Fig. 2. The effect of population size on the achieved results

In all the above-mentioned numerical experiments, the algorithm converges in less than 200 iterations; therefore, the termination criterion is set to this number since it guarantees the convergence. After 10 executions of the algorithm with the above-mentioned parameters, the best obtained sequence that has the lowest makespan for the instance problem is achieved as follows:

[1 2 3 1 2 3 4 4 1 2 3 4 1 2 4 3]

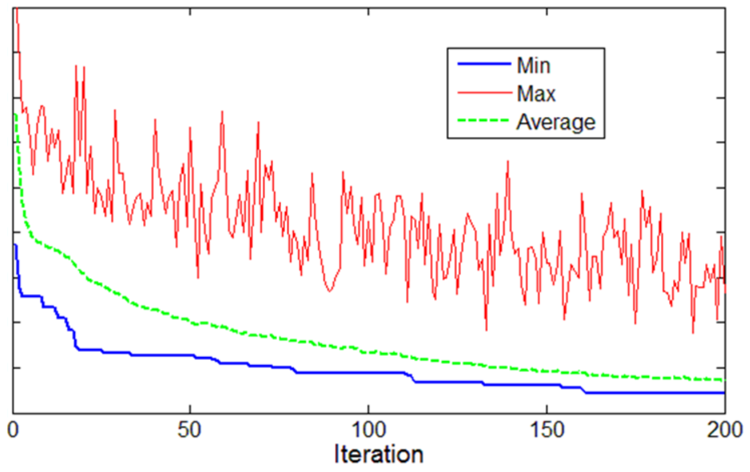

Fig. 3. The Process of evolution in DPSO to find a near-optimum solution

The evolution process in the proposed algorithm is shown in Fig. 2 where it can be seen that the variation of objective function, as it is expected, is drastic in the early stages of the optimization, less than 20 iterations, and continues throughout the process. The convergence, in this run, takes place in 162 iterations. However, since the algorithm is random-based, the convergence may occur in different stages of the optimization and the results could be slightly different. Fig. 3 also depicts the variation of maximum and average values of all solutions in each iteration, it is seen that the average value has the same trend as the achieved minimum result, though its variations continue even after convergence happens. On the other hand, the changes in the maximum value of the solutions are radical throughout the optimization course. This behavior could be related to the diverse nature of the proposed algorithm.

The presented results for the algorithm are the best out of 10 executions of the computer code. The presented convergence rate is the number of iterations in which the algorithm reaches a near-optimum solution. The presented convergence rate for the algorithm is the average of 10 executions. The computational time is also presented. The algorithm is programmed with MATLAB and is run on a computer with intel CORE i7 CPU and 8G RAM. The presented computational time is also the average of 10 executions.

## V. Conclusions

In this paper a novel evolutionary approach based on PSO is employed for solving the problem of schedule of a registration problem in a university. Minimizing the makespan in a real-world case study is considered as the target function. First, the proposed algorithm is employed for a case study that includes 4 students with 4 operators to show the characteristics of the problems. Afterwards the same procedure applied for the real-world case-study which is included registration of 167 students. The results indicate that the discrete version of PSO can find a near-optimum schedule with higher accuracy in less computational time in this case. The proposed solution for this problem is simple in concept and can easily be implemented in any registration problem in different sizes without much additional effort. The proposed approach to solve this problem is simple in concept and can easily be implemented in any registration problem in different sizes without much additional effort. In any registration problem with the same characteristics and different size the proposed particle can be applied and the procedure of discrete particle swarm optimization would be the same.

## References

[1] M. Yousefi and R. M. Yusuff, "Minimizing Earliness and Tardiness Penalties in a Single Machine Scheduling Against Common Due Date using Genetic Algorithm," *Res. Appl. Serv.*, vol. 4, no. 9, pp. 1205–1210, 2012.

[2] Y. Zheng, L. Lian, Z. Fu, and K. Mesghouni, "Evolutional Algorithm in Solving Flexible Job Shop Scheduling Problem with Uncertainties," in *In LISS 2013*, Springer Berlin Heidelberg, 2015, pp. 1009–1015.

[3] A. S. Manne, "On the job-shop scheduling problem," *Oper. Res.*, vol. 8, no. 2, pp. 219–223, 1960.

[4] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *J. Soc. Ind. Appl. Math.*, vol. 10, no. 1, pp. 196–210, 1962.

[5] A. Ponsich and C. A. C. Coello, "A hybrid differential evolution-tabu search algorithm for the solution of job-shop scheduling problems," *Appl. Soft Comput.*, vol. 13, no. 1, pp. 462–474, 2013.

[6] P. Brucker, B. Jurisch, and B. Sievers, "A branch and bound algorithm for the job-shop scheduling problem," *Discret. Appl. Math.*, vol. 49, no. 1, pp. 107–127, 1994.

[7] M. Yousefi, M. Yousefi, and A. N. Darus, "A modified imperialist competitive algorithm for constrained optimization of plate-fin heat exchangers," *Proc. Inst. Mech. Eng. Part A J. Power Energy*, vol. 226, no. 8, pp. 1050–1059, 2012.

[8] M. Yousefi and R. M. Yusuff, "Minimising earliness and tardiness penalties in single machine scheduling against common due date using imperialist competitive algorithm," *Int. J. Prod. Res.*, vol. 51, no. 16, pp. 4797–4804, 2013.

[9] S. Meeran and M. S. Morshed, "A hybrid genetic tabu search algorithm for solving job shop scheduling problems: a case study," *J. Intell. Manuf.*, vol. 23, no. 4, pp. 1063–1078, 2012.

[10] S. S. Panwalkar and W. Iskander, "A survey of scheduling rules," *Oper. Res.*, vol. 25, no. 1, pp. 45–61, 1977.

[11] T.-C. Chiang and L.-C. Fu, "Using dispatching rules for job shop scheduling with due date-based objectives," *Int. J. Prod. Res.*, vol. 45, no. 14, pp. 3245–3262, 2007.

[12] J. C. Tay and N. B. Ho, "Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems," *Comput. Ind. Eng.*, vol. 54, no. 3, pp. 453–473, 2008.

[13] J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Manage. Sci.*, vol. 34, no. 3, pp. 391–401, 1988.

[14] E. Demirkol, S. Mehta, and R. Uzsoy, "A computational study of shifting bottleneck procedures for shop scheduling problems," *J. Heuristics*, vol. 3, no. 2, pp. 111–137, 1997.

[15] C.-L. Chen and C.-L. Chen, "Bottleneck-based heuristics to minimize total tardiness for the flexible flow line with unrelated parallel machines," *Comput. Ind. Eng.*, vol. 56, no. 4, pp. 1393–1401, 2009.

[16] R. Zhang and C. Wu, "A hybrid approach to large-scale job shop scheduling," *Appl. Intell.*, vol. 32, no. 1, pp. 47–59, 2010.

[17] J. F. Gonçalves, J. J. de Magalhães Mendes, and M. G. C. Resende, "A hybrid genetic algorithm for the job shop scheduling problem," *Eur. J. Oper. Res.*, vol. 167, no. 1, pp. 77–95, 2005.

[18] R. Zhang and C. Wu, "A simulated annealing algorithm based on block properties for the job shop scheduling problem with total weighted tardinessobjective," *Comput. Oper. Res.*, vol. 38, no. 5, pp. 854–867, 2011.

[19] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, 1997, vol. 5, pp. 4104–4108.