International Journal of Operations Research Vol. 16, No. 2, 37-44 (2019)

An Improved Differential Evolution Algorithm for Permutation Flow Shop Scheduling Problem

Rudi Nurdiansyah[1*], Ong Andre Wahju Rijanto[2], Budi Santosa[3], and Stefanus Eko Wiratno[3]

[1]Industrial Engineering, Universitas Negeri Malang,
Jalan Semarang 5, Malang, Indonesia 65145

[2]Department of Industrial Engineering, Universitas Wijaya Putra,
Jalan Raya Benowo 1-3, Surabaya, Indonesia 60195

[3]Department of Industrial Engineering, Institut Teknologi Sepuluh Nopember,
Jalan Raya ITS, Keputih, Surabaya, Indonesia 60111

Abstract: The permutation flow shop scheduling problem (PFSP) widely arise in many industrial manufacturing systems. The optimal schedule for this problem is very important for a competitive position. Many methods are developed to generate more effective and efficient scheduling. An improved differential evolution (DE) algorithm is proposed to solve PFSP with the objective of minimizing makespan and total flowtime. Several developments are proposed to improve the DE's performance. We control the value of DE's parameter using adaptive parameters to increase the variety of the population. The crossover operator is modified to decide whether the individuals are mutated or not. Then, we employ local search to increase the quality solution of DE. The proposed algorithm is called iDEAMCO and tested using well-known problems in literature. The average relative errors of iDEAMCO in terms of makespan, total flowtime, and multi objectives are 0.618, 0.128, and 0.472, respectively, which outperform HAMC, GA, and MOACSA.
Keyword — Permutation flow shop, Multi objective, Meta-heuristics

1. INTRODUCTION

The improvements of the scheduling may take significant benefits in the production planning and manufacturing systems for a competitive position such as reducing the processing time and increasing the production efficiency and profits (Li & Yin, 2013; Lin, Gao, Li, & Zhang, 2015). One of the typical scheduling problems is the permutation flow shop scheduling problem (PFSP) that exists in modern industries such as food and glass industries. PFSP is a combinatorial optimization problem and has been extensively investigated by many researchers. In PFSP, a group of m machines processes a set of n jobs. Each of the n jobs is processed on m machines in the same order. One machine processes at most one job at a time and each job is processed on at most one machine at a time. The processing of each job is without pre-emption (Hejazi & Saghafian, 2005). The common objective of PFSP is to minimize the total completion time i.e. makespan.

Most studies on PFSP focus on the minimizing of makespan. In fact, there is another objective such as minimizing total flowtime which is also an essential objective to reduce the production cost (Yagmahan & Yenisey, 2010). While the minimizing of makespan leads to total production run utilization, the minimizing of total flowtime makes the stability of resources consumption, rapid turn-around of jobs and minimizing the work-in-process inventory (Yagmahan & Yenisey, 2010). Therefore, this research considers two objectives simultaneously i.e. the minimizing of makespan and total flowtime in order to reduce the production cost.

Qiu and Wang (2010) classify the methods to solve PFSP into three categories i.e. the precisions methods, constructive methods, and meta-heuristic algorithms. The precision methods such as enumeration method, cutting plane, and branch and bound require a high computation complexity. They are only suitable for a small scale of PFSPs. The constructive methods including Palmer Method (Palmer, 1965), Campbell-Dudek-Smith (CDS) method (Campbell, Dudek, & Smith, 1970), Johnson method (Garey, Johnson, & Sethi, 1976), Nawaz-Enscore-Ham (NEH) method (Nawaz, Jr, & Ham, 1983), Gupta method (Gupta, Shanthikumar, & Szwarc, 1987), Rajendran (RA) method (Gangadharan & Rajendran, 1993) can quickly obtain the solutions but their solutions are only local optimal, not global

[*]Corresponding author's e-mail: rudi.nurdiansyah.ft@um.ac.id

http://doi.org/10.6886/IJOR.201906_16(2).0002

optimal (Lin et al., 2015). The meta-heuristic algorithms are proposed by researchers in recent years to solve PFSP since they have better solutions than other categories. For example genetic algorithm (GA) (Reeves, 1995; Zhang, Li, & Wang, 2009); simulated annealing (SA) algorithms (Osman & Potts, 1989; Seyed-Alagheband, Davoudpour, Doulabi, & Khatibi, 2009); ant colony optimization (ACO) algorithm (Yagmahan & Yenisey, 2010; Ying & Liao, 2004); artificial bee colony (ABC) algorithm (Tasgetiren, Pan, Suganthan, & Chen, 2011); tabu search (TS) algorithm (Grabowski & Wodecki, 2004); particle swarm optimization (PSO) algorithms (Tasgetiren, Liang, Sevkli, & Gencyilmaz, 2007); and artificial immune systems (Gao & Liu, 2007).

One of meta-heuristic algorithms that become an effective global optimization method is the Differential Evolution (DE) algorithm. Besides having good performance for solving a variety of optimization problems, the special qualities of DE are the simplicity of the concept, easy implementation, and fast converging (Das, Mullick, & Suganthan, 2016; Storn & Price, 1997). Pan, Tasgetiren, and Liang (2008); Tasgetiren, Liang, Sevkli, and Gencyilmaz (2004); Qian et al. (2008); and Mingyong and Erbao (2010) have proven that DE has the capability to solve a combinatorial optimization problem and obtain a better performance than other algorithms.

DE has four main steps i.e. initialization, mutation, crossover, and selection. DE iteratively evaluates the function evaluations of the problem using mutation, crossover, and selection to improve the obtained solution during the search. In the initialization, the target population is generated. Next, the target population is mutated by a mutant factor to form the mutant population. In forming the mutant population, there is a scaling factor (F) that controls the growth rate of the mutant population. Furthermore, the crossover operator combines the mutant population with the target population to generate the experimental population. The crossover operator in combining the mutant population with the target population depends on a crossover rate (Cr). The selection operator compares the fitness value function between the experimental population and the target population. In the end, the best individuals become a member of the next generation. These procedures are repeated until the stopping criterion is reached.

It is noticed that the performance of DE depends on the settings of its parameters i.e. the scaling factor F and crossover rate C_r (J. Liu, 2002; Mingyong & Erbao, 2010; Qian et al., 2008; Tvrdík, 2007). Zaharie (2002); Tvrdík (2007); Thangaraj, Pant, and Abraham (2009); and Mingyong and Erbao (2010) control both parameters using a certain formula so that in every generation, the value of F and C_r is iteratively changed to increase the variety of the population. This procedure is called adaptive parameters. Furthermore, to enhance the quality solution of DE, Noman and Iba (2008); Sauer and Coelho (2008); Pan et al. (2008); and Zamuda, Brest, Boskovic, and Zumer (2009) combine DE with local search procedure. Noman and Iba (2008) have proven that DE combined with local search procedure has a better solution than original DE.

This study develops the DE algorithm using adaptive parameters to increase the variety of the population and combine DE with local search to improve the quality solution of the found solution in every iteration. Moreover, the crossover operator is modified to simplify the procedure of DE. The original DE combines the mutant population and target population to generate the experimental population. In this study, the crossover operator selects the individuals in the target population to be mutated. Therefore, the crossover operator is performed before the mutation step.

The contribution of this research is that a new variant of DE, named **iDEAMCO**, improved DE with adaptive parameter and modified crossover operator, is proposed to solve the multi objectives PFSP. We test the proposed algorithm using well-known problems in literature and compare its performance with the HAMC algorithm proposed by Ravindran, Selvakumar, Sivaraman, and Haq (2005), GA and MOACSA (multi objectives ant colony system algorithm) proposed by Yagmahan and Yenisey (2010). The experimental results demonstrate that iDEAMCO has a better performance than HAMC, GA, and MOACSA with respect to makespan, total flowtime, and multi objectives.

The rest of the paper is organized as follows. Section 2 describes the mathematical model. Section 3 introduces the proposed algorithm, called iDEAMCO. Section 4 illustrates the computational experiments. Section 5 concludes and suggests future research.

2. PROBLEM FORMULATION

The multi objectives PFSP consists of the scheduling of n jobs with given processing time on m machines. The assumptions of PFSP are m machines process n jobs in the same sequence, each machine processes only one job at any time, one job can be processed only on one machine at any time, and no pre-emption during processing each job (Y. F. Liu & Liu, 2013). We use the following notations:

n	the number of jobs
m	the number of machines
S	an arbitrary sequence of n jobs
π_j	the job processing sequence of job j

$P(j,k)$	the processing time of job j on machine k
$C(\pi_j,k)$	the completion time of job j on machine k
w_1	weight associated with makespan $(0 \leq w_1 \leq 1)$
w_2	weight associated with flowtime $(0 \leq w_2 \leq 1)$
$M(S)$	the makespan criterion of S
$F(S)$	the total flowtime criterion of S
$TMF(S)$	multi objective function

The mathematical model of this problem is as follows (Y. F. Liu & Liu, 2013):
Decision variants: $\pi = \{\pi_1, \pi_2, \cdots, \pi_n\}$
Subject to:

$$\begin{aligned}
&C(\pi_1,1) = P(1,1)\\
&C(\pi_j,1) = C(\pi_{j-1},1) + P(j,1), \quad j = 2,3,\cdots,n\\
&C(\pi_1,k) = C(\pi_1,k-1) + P(1,k), \quad k = 2,3,\cdots,m\\
&C(\pi_j,k) = \max(C(\pi_{j-1},k), C(\pi_j,k-1)) + P(j,k), \quad j = 2,3,\cdots,n; k = 2,3,\cdots,m
\end{aligned} \tag{1}$$

Our two objectives are as follows:
(1) Minimizing makespan: $M(S) = \min C_{\max} = C(\pi_j, m)$.
(2) Minimizing total flowtime: $F(S) = \min TFT = \sum_{j}^{n} C(\pi_j, m)$.

The weighted combination of makespan and total flowtime of S:

$$TFT(S) = w_1 \cdot M(S) + w_2 \cdot F(S) \tag{2}$$

3. PROPOSED ALGORITHM

In this section, the proposed algorithm, called iDEAMCO, is introduced and explained. DE is firstly used by Storn and Price (1997) to solve the Chebychev polynomial fitting problem and proven to be effective. Different from the original DE, several improvements are used in this study to improve the performance of DE. First, we adapt the scaling factor F and crossover rate C_r by calculating both parameters with a certain formula in every iteration so that the value of both parameters is iteratively changed to improve the variance of the population. Second, to simplify the procedure of DE, we modify the crossover operator to be the operation to choose which individuals in the target population to be mutated so that the crossover operator is performed before mutation. Third, we combine DE with local search procedure to avoid our algorithm being trapped in the local optimal. The steps of iDEAMCO to solve PFSP are as follows:

3.1 Initialization

Before performing the initialization, it is necessary to determine the upper bound (ub) and lower bound (lb). The number of population is denoted by NP. For the initial generation $g = 0$, a population of NP random individuals is initialized by the following equation:

$$X_{j,i,g_0} = lb_{j,i} + rand(0,1)(ub_{j,i} - lb_{j,i}), \quad j = 1,2,\cdots,n; i = 1,2,\cdots,m \tag{3}$$

where i is the index of an individual X_i, $rand(0,1)$ is a random number between 0 and 1. This step generates continuous numbers.

3.2 Crossover

The original DE uses the crossover operator to combine the target population and mutant population to create the experimental population. In this study, the crossover operator is used to choose which individuals in the target population to be mutated. In this step, random numbers are generated. If the random number is equal or less than the crossover rate C_r, then this individual is chosen to be mutated in the mutation step. Otherwise, this population is not chosen to be mutated. The formula used to calculate the crossover rate in every iteration is as follows (Mingyong & Erbao, 2010):

$$C_r = C_r \min + G\frac{C_r \max - C_r \min}{MAXGEN} \tag{4}$$

C_r min and C_r max are the minimum and the maximum value of crossover rate C_r, respectively. G is the current iteration, while MAXGEN is the maximum number of iterations. As stated by Mingyong and Erbao (2010), this formula is used to improve the variance of the population and avoid DE being trapped in the local optimal.

3.3 Mutation

This step mutates the populations chosen by the crossover operator. The mutation is done by adding the difference of two individuals (taken randomly) to a third individual by:

$$V_{i,g} = X_{r_1,g} + F(X_{r_2,g} - X_{r_3,g}) \tag{5}$$

where r_1, r_2, and r_3 are chosen from the current population. The scaling factor $F \in (0, 1)$ is to control the rate of population growth. In this step, the scaling factor F iteratively changes using the following equation (Tvrdík, 2007):

$$F = \begin{cases} \max\left(F\min, 1 - \left|\frac{f\max}{f\min}\right|\right), & if \quad \left|\frac{f\max}{f\min}\right| < 1 \\ \max\left(F\min, 1 - \left|\frac{f\min}{f\max}\right|\right), & otherwise \end{cases} \tag{6}$$

where f min and f max are the minimum and maximum fitness value of the population, respectively. F min is the input parameter to ensure $F \in (F\min, 1)$. This formula illustrates a diverse and intensive search to achieve global optimal solution (Tvrdík, 2007).

3.4 Insert-Based Local Search

In this study, we use insert-based local search to improve the quality solution of iDEAMCO. Tasgetiren et al. (2004) state that insert-based local search has a more efficient and thorough search than other neighborhoods search such as *interchange* and *swap* with the same computational efforts. The procedure of the insert-based local search is given as follows: choose randomly two jobs in a job permutation π_j. Denote these two jobs, say u and v. Insert the job u to the position of job v. Then, evaluate the objective function. This procedure is terminated if the current best objective function is not improved.

3.5 Selection

If $V_{i,g}$ has a smaller objective function than $X_{i,g}$, then $V_{i,g}$ replaces $X_{i,g}$ in the next generation. Otherwise, $X_{i,g}$ remains in the population.

$$X_{i,g+1} = \begin{cases} V_{i,g}, & if \quad f(V_{i,g}) \leq f(X_{i,g}) \\ X_{i,g}, & otherwise. \end{cases} \tag{7}$$

3.6 Stopping Criterion

The stopping criterion of the algorithm is the maximum number of iterations.

The pseudocode of iDEAMCO for PFSP is presented in **Algorithm 1**. After the population is initialized, convert each individual defined by the continuous numbers to the job permutation using SPV (Smallest Position Value). Evaluate each individual by computing the objective function using (2). At the beginning of the optimization loop, convert the job permutation back to the individual $X_{j,i}$. Perform crossover to choose the individual required to be mutated. Here, we calculate the crossover rate C_r using (4). Then, mutate the individual chosen by the crossover operator using (5). (6) computes the scaling factor F in this iteration. Apply insert-based local search by choosing two jobs in the position, say u and v, and insert the job u to v. Perform selection and update the population. Go back to the crossover. Perform crossover, mutation, selection, and local search procedure iteratively until the stopping criterion is reached. At the end of the iteration, the best objective function is obtained.

Algorithm 1: iDEAMCO for PFSP

```
Step 1 : Input number of population $NP$, $F$ min and $C_r$ min, $C_r$ max, number of iteration $g$ max. Set
         $S = \varnothing$ and let the initial $lb = -1$, $ub = 1$.
Step 2 : Population initialization. Generate
         $X_{j,i,0} = lb(X_{j,i}) + random(0,1)(ub(X_{j,i}) - lb(X_{j,i})), j = 1, \cdots, n$ for $i = 1, \cdots, m$.
Step 3 : Convert individual $X_{j,i}$ to a job permutation $\pi_i$ according to the SPV rule. Evaluate every
         individual $\pi_i$ using $TMF(S)$'s formula.
Step 4 : Let $g = 1$.
Step 5 : Convert $\pi_i$ back to $X_{j,i}$
         Perform DE's *Crossover* and *Mutation.*
         **If** $random(0,1) \leq C_r$, **then**
             Randomly select $r_1, r_2, r_3 \in (1, \cdots, m)$, where $r_1 \neq r_2 \neq r_3 \neq i$.
             Let $V_{j,i}(g) = X_{r_1,j,i} + F(X_{r_2,j,i} - X_{r_3,j,i})$
         **Else** $V_{j,i}(g) = X_{j,i}(g)$
         **End if**
Step 6 : Apply insert-based local search
         **Step 6.1:** Convert individual $V_{i,g}$ to a job permutation $\pi_{i_0}$.
         **Step 6.2:** Set $loop = 1$;
             **Do**
             Randomly select $u$ and $v$, where $u \neq v$;
             $\pi_{i_1} = Insert(\pi_{i_0}, u, v)$;
             **If** $f(\pi_{i_1}) < (\pi_{i_0})$, **then** $\pi_{i_0} = \pi_{i_1}$;
             **Else** go to **Step 6.3**
             **End if**
             $loop{+}{+}$;
         **Step 6.3:** If $f(\pi_{i_1}) > (\pi_{i_0})$, then $\pi_{i_0}$ remains the same;
Step 7 : Perform Selection. If $\pi_{i_0} \leq \pi_i$, then $\pi_i = \pi_{i_0}$. Else, $\pi_i = \pi_i$.
Step 8 : Update $S$.
Step 9 : Let $g = g + 1$. If $g < g$ max, then go to **Step 5**.
Step 10: Output $S$ and its objective value.
```

4. COMPUTATIONAL EXPERIMENTS

In this section, the results of computational experiments are presented to evaluate the performance of iDEAMCO. The proposed algorithm is tested on 28 benchmark problems with 20 jobs and the number of machines varying from 5 to 20 given by Taillard (1993). The computer programs of the proposed algorithm are developed in Matlab 7.8 and implemented on Intel Core Duo 1.66 GHz system with 1024 MB DDR-2 RAM. The performance of iDEAMCO is compared with HAMC algorithms (HAMC1, HAMC2, HAMC3), GA, and MOACSA. The parameters setting of iDEAMCO in this study are defined as follows: the number of populations $= 100$; C_r min $= 0.3$; C_r max $= 0.9$; F min $= 0.5$; and number of iterations $= 1000$. The steps of GA are initialization, evaluation, selection, crossover, mutation, elitist strategy, and stopping criterion with the parameters setting are as follows: population size $= 20$; initialization $=$ randomly generated; number of iterations $= 60$; crossover rate $= 0.8$; mutation rate $= 0.2$; crossover operator $=$ Goldberg's partially mapped crossover (MPX) operator; mutation operator $=$ shift change.

Every test is repeated with 10 runs for each instance and the best solution is selected. Hence, there are 280 runs in total. The weight of makespan and total flowtime is the same i.e. $w_1 = w_2 = 0.5$ for multi objectives. The relative error (RE) in total objective value for schedule S generated by any algorithms is given as follows:

$$RE(S) = w_1 \cdot \left(\frac{M(S) - \min(M(S))}{\min(M(S))}\right) + w_2 \cdot \left(\frac{F(S) - \min(F(S))}{\min(F(S))}\right) \tag{8}$$

The average relative errors obtained by makespan, total flowtime, and multi objectives are presented in Figure 1-3, respectively. The average relative error is obtained from the sum of the relative errors of every problem divided by the number of the problem i.e. 28. HAMC1, HAMC2, and HAMC3 express HAMC algorithm. GA, MOACSA, and iDEAMCO express genetic algorithm, multi objectives ant colony system algorithm, and the proposed method, respectively. The vertical axis in Figure 1-3 shows the average relative error.

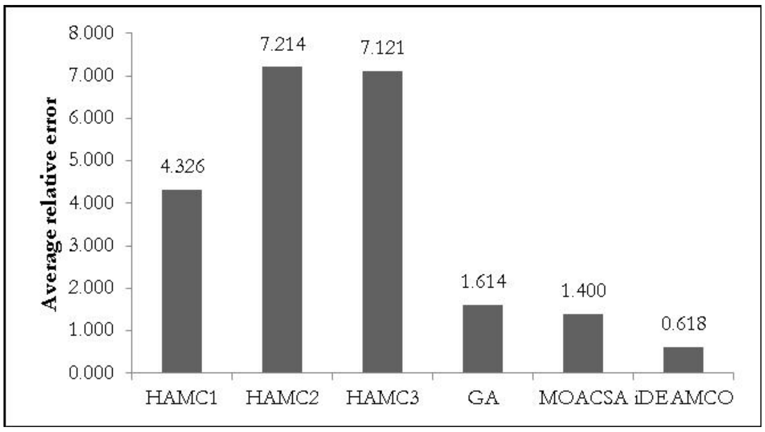

Figure 1: Average relative error in makespan

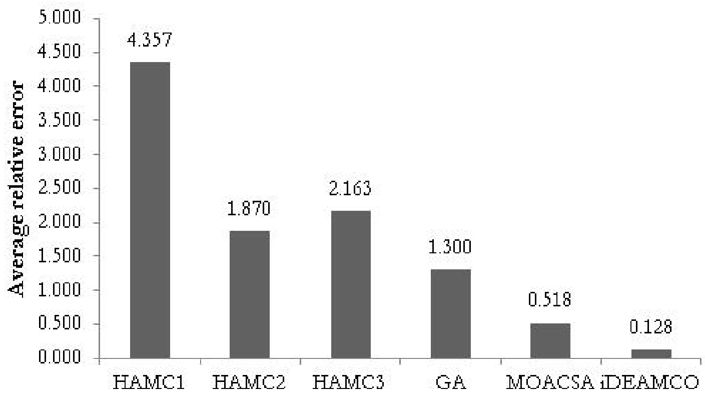

Figure 2: Average relative error in total flowtime

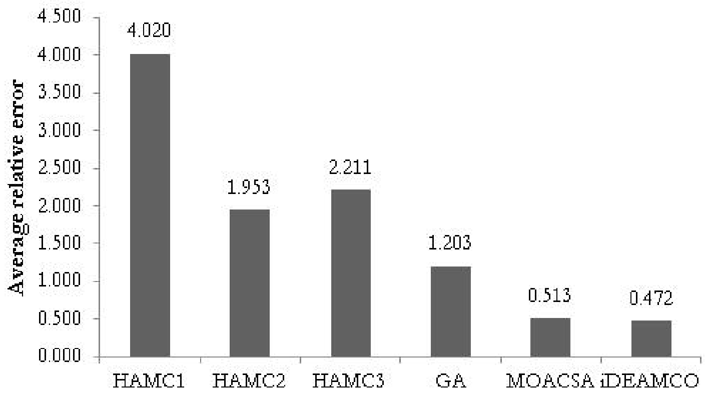

Figure 3: Average relative error in multi objectives

The results show that iDEAMCO outperforms HAMC1, HAMC2, HAMC3, GA, and MOACSA in makespan, total flowtime, and multi objectives. The average relative errors of iDEAMCO with respect to the makespan, total flowtime, and multi objectives are 0.618, 0.128, and 0.472, respectively. These results are better than other methods and illustrate that iDEAMCO is able to improve the quality solution of the DE algorithm.

5. CONCLUSION AND FUTURE RESEARCH

In this paper, iDEAMCO is proposed to solve multi objectives permutation flow shop scheduling problem with makespan and total flowtime criteria. iDEAMCO is based on the development of the DE algorithm. There are several improvements to the DE algorithm in this study. First, adaptive parameters of scaling factor F and crossover rate C_r are applied to increase the variance of the population. Second, the crossover operator is modified to simplify DE's procedure. The last, an insert-based local search as a neighborhood search is employed to avoid our proposed algorithm being trapped in local optimal. Experimental results and comparisons demonstrate the effectiveness of iDEAMCO. The results of this research indicate that iDEAMCO has a better performance than HAMC1, HAMC2, HAMC3, GA, and MOACSA in makespan, total flowtime, and multi objectives. The contribution of this study is that a new variant of DE, namely iDEAMCO, outperforms other state-of-the-art algorithms from the literature for solving multi objectives PFSP. In our future work, we will analyze the convergence property of iDEAMCO, develop effective DE for other kinds of scheduling problems such as job shop scheduling, flexible manufacturing, open shop scheduling, etc. and incorporate a suitable way into DE to propose powerful algorithms for stochastic scheduling problems.

REFERENCES

Campbell, H. G., Dudek, R. A., & Smith, M. L. (1970). A Heuristic Algorithm for the n Job, m Machine Sequencing Problem. *Management Science*, *16*(10), 630-637.

Das, S., Mullick, S. S., & Suganthan, P. N. (2016). Recent advances in differential evolution - An updated survey. *Swarm and Evolutionary Computation*, *27*, 1-30.

Gangadharan, R., & Rajendran, C. (1993). Heuristic algorithms for scheduling in the no-wait flowshop. *International Journal of Production Economics*, *32*(3), 285-290.

Gao, H., & Liu, X. (2007). Improved Artificial Immune Algorithm and its application on the Permutation Flow Shop Sequencing Problems. *Information Technology Journal*, *6*(6), 929-933.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, *1*(2), 117-129.

Grabowski, J., & Wodecki, M. (2004). A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. *Computers & Operations Research*, *31*(11), 1891-1909.

Gupta, J. N., Shanthikumar, J. G., & Szwarc, W. (1987). Generating improved dominance conditions for the flowshop problem. *Computers & operations research*, *14*(1), 41-45.

Hejazi, S. R., & Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: a review. *International Journal of Production Research*, *43*(14), 2895-2929.

Li, X., & Yin, M. (2013). A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem. *International Journal of Production Research*, *51*(16), 4732–4754.

Lin, Q., Gao, L., Li, X., & Zhang, C. (2015). A hybrid backtracking search algorithm for permutation flow-shop scheduling problem. *Computers & Industrial Engineering*, *85*, 437-446.

Liu, J. (2002). On setting the control parameter of the differential evolution method. In *Proceedings of the 8th international conference on soft computing (mendel 2002)* (p. 11-18).

Liu, Y. F., & Liu, S. Y. (2013). A hybrid discrete artificial bee colony algorithm for permutation flowshop scheduling problem. *Applied Soft Computing*, *13*(3), 1459-1463.

Mingyong, L., & Erbao, C. (2010). An improved differential evolution algorithm for vehicle routing problem with simultaneous pickups and deliveries and time windows. *Engineering Applications of Artificial Intelligence*, *23*(2), 188-195.

Nawaz, M., Jr, E. E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, *11*(1), 91-95.

Noman, N., & Iba, H. (2008). Accelerating differential evolution using an adaptive local search. *IEEE Transactions on Evolutionary Computation*, *12*(1), 107-125.

Osman, I. H., & Potts, C. N. (1989). Simulated annealing for permutation flow-shop scheduling. *Omega*, *17*(6), 551-557.

Palmer, D. S. (1965). Sequencing Jobs Through a Multi-Stage Process in the Minimum Total Time—A Quick Method of Obtaining a Near Optimum. *Journal of the Operational Research Society*, *16*(1), 101-107.

Pan, Q. K., Tasgetiren, M. F., & Liang, Y. C. (2008). A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Computers & Industrial Engineering*, *55*(4), 795-816.

Qian, B., Wang, L., Hu, R., Wang, W.-L., Huang, D.-X., & Wang, X. (2008). A hybrid differential evolution method for permutation flow-shop scheduling. *The International Journal of Advanced Manufacturing Technology*, *38*(7-8), 757-777.

Qiu, C. H., & Wang, C. (2010). An immune particle swarm optimization algorithm for solving permutation flowshop problem. *Key Engineering Materials*, *419-420*, 133-136.

Ravindran, D., Selvakumar, S. J., Sivaraman, R., & Haq, A. N. (2005). Flow shop scheduling with multiple objective of minimizing makespan and total flow time. *The international journal of advanced manufacturing technology*, *25*(9-10), 1007-1012.

Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. *Computers & operations research*, *22*(1), 5-13.

Sauer, J. G., & Coelho, L. S. (2008). Discrete Differential Evolution with local search to solve the Traveling Salesman Problem: Fundamentals and case studies. In *2008 7th ieee international conference on cybernetic intelligent systems* (p. 1-6).

Seyed-Alagheband, S. A., Davoudpour, H., Doulabi, S. H. H., & Khatibi, M. (2009). Using a modified simulated annealing algorithm to minimize makespan in a permutation flow-shop scheduling problem with job deterioration. In *Proceedings of the world congress on engineering and computer science 2009 vol ii* (p. 20-22).

Storn, R., & Price, K. (1997). Differential Evolution - A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, *11*(4), 341-359.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*(2), 278-285.

Tasgetiren, M. F., Liang, Y.-C., Sevkli, M., & Gencyilmaz, G. (2004). Differential evolution algorithm for permutation flowshop sequencing problem with makespan criterion. In *Proceedings of the 4th international symposium on intelligent manufacturing systems (ims 2004).*

Tasgetiren, M. F., Liang, Y.-C., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research*, *177*(3), 1930-1947.

Tasgetiren, M. F., Pan, Q.-K., Suganthan, P. N., & Chen, A. H.-L. (2011). A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops. *Information Sciences*, *181*(16), 3459-3475.

Thangaraj, R., Pant, M., & Abraham, A. (2009). A simple adaptive differential evolution algorithm. In *2009 world congress on nature & biologically inspired computing (nabic)* (p. 457-462).

Tvrdík, J. (2007). Differential evolution with competitive setting of control parameters. *Task Quarterly*, *11*(1-2), 169-179.

Yagmahan, B., & Yenisey, M. M. (2010). A multi-objective ant colony system algorithm for flow shop scheduling problem. *Expert Systems with Applications*, *37*(2), 1361-1368.

Ying, K. C., & Liao, C. J. (2004). An ant colony system for permutation flow-shop sequencing. *Computers & Operations Research*, *31*(5), 791-801.

Zaharie, D. (2002). Critical values for the control parameters of differential evolution algorithms. In *Proceedings of mendel 2002, 8th international mendel conference on soft computing* (p. 62-67).

Zamuda, A., Brest, J., Boskovic, B., & Zumer, V. (2009). Differential evolution with self-adaptation and local search for constrained multiobjective optimization. In *2009 ieee congress on evolutionary computation* (p. 195-202).

Zhang, Y., Li, X., & Wang, Q. (2009). Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization. *European Journal of Operational Research*, *196*(3), 869-876.