Hybrid Differential Evolution and Bottleneck Heuristic Algorithm to Solve Bi-Objective Hybrid Flow Shop Scheduling Unrelated Parallel Machines Problem

Riza A. Rahman, Budi Santosa, and Stefanus E. Wiratno
Department of Industrial Engineering
Sepuluh Nopember Institute of Technology
Surabaya, Indonesia

Abstract

This paper combines the differential evolution algorithm (DE) with bottleneck heuristic for two-objective minimization makespan and total tardiness on a hybrid flow shop scheduling with unrelated parallel machines (HFS-UPM). In this model, some machines at each stage are not identical (unrelated parallel machines) and have different processing time. This paper applies algorithm that based on combination DE algorithm with bottleneck based heuristic algorithm as an initialization to get a better non-dominated solution. DE random generated solution, opposition based learning solution and solution from bottleneck based heuristic combined by non-dominated sorting to get initial population. In this paper, we consider the problem with the objective of minimizing makespan and total tardiness solved by simultaneous non-dominated sorting and crowding distance approach. The comparisons of algorithm demonstrated the effectiveness of the proposed DE-hybrid algorithm. It is shown that the proposed DE-hybrid algorithm more effective than DE-ns algorithm and DE-based algorithm.

Keywords

Differential evolution, bottleneck based heuristic, makespan, total tardiness, hybrid flow shop unrelated parallel machines.

1. Introduction

Production scheduling is one of the critical issues in the planning and manufacturing process. Scheduling problem focused on how to allocate machines to perform a collection of activities in a period of time in order to optimize a certain objective (Pinedo, 2008).

Recently, most of research in production scheduling is concerned with the optimization of a single criterion. However, the analysis of the performance of a schedule often involves more than one criterion and therefore requires multi-objective analysis. Production scheduling needs to consider two or more objectives that can conflict with each other simultaneously. Multi-objective optimization that conflict with each other is focused on finding the solution set, where there is no single performance criteria that can be improved without sacrificing the quality of the performance of other criteria. This is commonly referred to as non-dominated solutions or pareto solutions (Ruiz and Vázquez-Rodríguez, 2010).

One type of production scheduling is still being developed is the hybrid flow shop scheduling. In industry, hybrid flow shop scheduling n-job and k-stage is the common manufacturing environment such as semiconductor, paper and textile industries, and chemical industries. Hybrid flow shop (HFS) also called flexible flow lines, flow shop with parallel machines, or multiprocessor flow shop (Quadt and Kuhn, 2007). According to Ruiz and Rodriquez (Ruiz and Vázquez-Rodríguez, 2010) the hybrid flow shop (HFS) is common manufacturing environment in which is a set of n jobs are to be processed in a series of m stages. All jobs are processed following the same production flow: stage 1, stage 2,..., stage m. A job might skip any number of the stage provided. A job is processed in at least one of them. In the HFS has at least two stages of operation and at least each stage has more than one unit machine in parallel. Some problems in the field, the machine at each stage are not identical (unrelated parallel machines) and have different processing time.

The scheduling problem with multi-objective optimization using metaheuristic has been applied by several researchers. Among them, Tasgetiren, et al. (Tasgetiren et al., 2007) using the hybridization of particle swarm optimization (PSO) algorithm with variable neighborhood search (VNS) for makespan and maximum lateness minimization in permutation flowshop scheduling. Li, et al. (Li and Yin, 2013) using the PSO-based hybrid algorithm to minimize makespan and maximum tardiness simultaneously. Jungwattanakit, et al. (Jungwattanakit et

al., 2009) using GA to minimize the makespan and the number of tardy jobs in flexible flow shop scheduling problem. Qian, et al.(Qian et al., 2009) using a hybrid differential evolution (DE)-based algorithm to minimize makespan and maximum tardiness simultaneously using the concept of pareto solutions in job flow shop scheduling with limited buffers. DE has emerged as one of the simple and efficient techniques for solving single objective global optimization problems in which it has shown a great robustness and a fast convergence (Vesterstrom and Thomsen, 2004). These are precisely the characteristics of DE that make it attractive to extend it to solve Multi Objective Problem. The aim of this paper is to present a new hybrid Differential Evolution (DE) algorithm for solving Multi-objective hybrid flow shop scheduling unrelated parallel machines.

2. Problem Formulation of HFS-UPM

2.1. Mathematical model

In this section, we provide a 0-1 mixed integer linier programming formulation for the problem under consideration.

- Notation

  j,h job index, $j,h = 1,2,...,n$
  i stage index, $i = 1,2,...,m$
  l parallel machine index, $l=1,2,....,k$
  C_{max} the makespan or maximum completion time
  T total tardiness
  C_{ij} the completion time of job j at stage i
  S_{ij} starting time of job j at stage i,
  n number of jobs to be scheduled
  m number of stage
  k_i number of parallel machines at stage i
  p_{ij} processing time of job j at stage i
  d_j due date of job j
  Y_{ihj} 1 if job h is scheduled immediately before job j on parallel machine l at stage i , and 0 otherwise
  X_{lij} 1 if job j scheduled at stage i at machine l, and 0 otherwise

- Mathematical Formulation

  The problem can be formulated as follows:

  Minimize: $C_{max} = max\ \{C_{ij}\}$, $T = \sum_{j=1}^{n} max\{C_{ij} - d_j, 0\}$ (1)

  Subject to:

  $C_{\max} \geq C_{ij}$, for all $s = 1, \ldots, k, j = 1, \ldots, n,$ (2)
  $C_{ij} = S_{ij} + \sum_{l=1}^{k} p_{lij} \times X_{lij},$ (3)
  $\sum_{l=1}^{k} X_{lij} = 1,$ for all $i = 1, \ldots, m, j = 1, \ldots, n,$ (4)
  $C_{ij} \leq S_{(i+1)j},$ for all $i = 1, \ldots, m - 1,$ (5)
  $S_{ih} \leq C_{ij} - MY_{ihj}$, for each pair of job (h, j), (6)
  $S_{ij} \leq C_{ih} - (1 - Y_{ihj})M,$ for each pair of job (h, j), (7)
  $X_{lij} \in \{0, 1\}, Y_{ihj} \in \{0, 1\}$
  For all $j = 1, \ldots, n, i = 1, \ldots, m$, dan $l = 1, \ldots, k.$ (8)

  The objective function (1) is to minimize the makespan and total tardiness with constraint (2) to ensure that the makespan is at least equal or bigger than completion times of the last job. Constraint (3) are used to the compute the completion time of job j at stage i. Constraint (4) ensures that each job is assigned/processed exactly by one machine at each stage. Constraints (5) ensure that each job can be started to process only when it has been completed at the precedent stage. Constraint (5), (6) and (7) ensure that only one job can be processed by each machine at one time and the starting time of the job must be greater than completion time of precedent job. The last constraint (8) defines binary values 0 or 1 to all decision variables.

- Model Assumption

  There are several assumptions that are commonly made regarding this problem :

  - All n job are independent and available to be processed at initial time.
  - The production stage has sufficient capacity to store and manage the WIP inventory generated during the execution of the complete set of jobs (infinite storage capacity at each stage).
  - One machine can process only one job at a time and one job can be processed by only one machine at any time.

- The ready times for all machines (times when the machines become available to process the set of jobs to be scheduled) are known.
- For all the jobs, the processing times at each stage are known and deterministic.
- Job set-up times are sequence-independent and are included in the job processing time at the corresponding stage.
- Travel time between consecutive stages is negligible. In instances where this assumption does not hold true, inter-stage travel can be treated as a processing step with the process time equal to the travel time.
- Pre-emption is not allowed, that is no interruption of a job processing is allowed.

3. Proposed Algorithm

This section will present each component of the proposed DE algorithm, including DE-based algorithm, multi-objective bottleneck heuristic, opposition based learning, non-dominated sorting and crowding distance, solution elimination and provided procedure of DE-hybrid algorithm at the last section.

3.1. Solution representation in DE

The encoding scheme based on job permutation has been widely used in permutation based genetic algorithm (GA) for flow shop scheduling problem. For example, for a three job flow shop problem, permutation [2 3 1] corresponds to the processing sequence from job-2, job-3 to job-1. However, every individual solution in DE is continuous value. So the continuous do not directly represent a schedule. The important issues in apply DE to HFS-UPM problem is to find a suitable mapping between job sequence and individual (continuous vector) in DE. In this paper, we propose smallest parameter value (SPV) rule to convert DE's individual X_i =[$x_{i,1}$, $x_{i,2}$,…,$x_{i,n}$] to the job permutation vector π_i = [$\pi_{i,1}$, $\pi_{i,2}$,..., $\pi_{i,n}$].

According to SPV rule, X_i =[$x_{i,1}$, $x_{i,2}$,…,$x_{i,n}$] are firstly ranked by ascending order to get some sequence. A simple example with six jobs is provided to show the SPV rule in Table 1. In the instance, the solution is X_i =[0.24, 0.56, 0.31, 0.97, 0.66, 0.18]. Because $x_{i,6}$=0,18 is the smallest value of X_i , $x_{i,6}$ is selected firstly and assigned rank value 1, then $x_{i,1}$ is selected secondly and assigned rank value 2. With the same way $x_{i,3}$, $x_{i,2}$, $x_{i,5}$ and $x_{i,4}$ are assign rank value 3,4,5 and 6, respectively. Thus, the sequence is π_i=[6 1 3 2 5 4].

Table 1: Solution representation

Dimension	1	2	3	4	5	6
X_{ij}	0.24	0.56	0.31	0.97	0.66	0.18
φ_{ij}	2	4	3	6	5	1
π_{ij}	6	1	3	2	5	4

3.2. DE-ns Algorithm

In this section, we will present the hybridization of DE for two-objective HFS-UPM. The procedure of DE-ns is summarized as follows:

Step 1, Let *g* denote a generation,P_g a target population with size *NP* in generation *G*, $X_{i,G}$ the *i*th individual with dimension *N* (*N*=*n*) in P_g. $x_{i,j,g}$ the *j*th variable of individual $X_{i,g}$. *CR* the crossover probability, *F* the fraction of mutation , and random(0,1) the random value in the interval [0,1]. Where *NP* is the number of population, *i*=1,2,...,*NP*. *Xmax* the upper limit, *Xmin* the lower limit. *Gmax* the maximum generation or iteration.

- Input the parameter value *N*,*PS*,*CR*∈[0,1],*F*∈[0,1], *Gmax*. Let *Xmax*=4, *Xmin*=0.

Step 2, initialization:

- Generate $X_{i,j\,0}$ = random (0,1) * ($Xmax_{i,j}$−$Xmin_{i,j}$) + $Xmin_{i,j}$.
- Convert $X_{i,j\,0}$ being an individual job permutation using smallest parameter value (SPV) rule. The individual job permutation will be member of initial target population, P_0.

Step-3, Calculate the two-objective function, makespan and total tardiness of each individual initial population

Step-4, update iteration: Set *G* = *G* +1

Step-5, generate mutant population:

For each target individual, $X_{i,G}$ at generation *G*, a mutant individual,$V_{i,G+1}$= [$V_{i,1,G+1}$,$V_{i,2,G+1,\ldots}$,$V_{i,n,G+1}$] is determined using DE/*rand*/1/*bin* scheme. This scheme give best performance for makespan, total tardiness and flowtime criteria (Onwubolu and Davendra, 2006). It is calculated by the following:

$$V_{i,} = X_{r1,i} + F\,(X_{r2,i} - X_{r3,i}) \quad (9)$$

Where Xr_{1i}, Xr_{2i}, Xr_{3i} are three individual randomly choosen from the populations such that $a_i \neq b_i \neq c_i$. $F > 0$ is a mutant factor.

Step-6, generate trial population

Following the mutation phase, the crossover (recombination) operator is applied to obtain the trial population. For each mutant individual, $V_{i,g+1}$ an integer random number between 1and n, i.e, $D_i \in (1,2,..,n)$ is chosen, and a trial individual, $U_{i,g} = [U_{i,1,g},...,U_{i,n,g}]$ for each target vector $X_{i,g}$ such that:

$$u_{ij}^{t+1} = \begin{cases} v_{ij}^{t+1}, if & r_{ij}^{t+1} \leq CR \quad or \quad j = D_i \\ x_{ij}^{t}, & Otherwise \end{cases} \quad (10)$$

where the index D refers to a randomly chosen dimension ($j=1,2,..,n$), which is used to ensure that at least one parameter of each trial individual $U_{i,g+1}$ differs from its counterpart in the previous generation $U_{i,g}$. CR is a user-defined crossover constant in the range [0, 1], and $r_{i,,j,g+1}$is a uniform random number between 0 and 1. In other words, the trial individual is made up with some parameters of mutant individual, or at least one of the parameters randomly selected, and some other parameters of target individual.

Step-7, the evaluation trial population

- Calculate the two-objective function of each individual trial population,$U_{i,g+1}$

Step 8, elimination

Both population target and trial are combined then eliminate solution that have same permutation job and leave one solution remains in the population.

Step-9, selection between target population ($X_{i,G}$) and trial populations ($U_{i,G+1}$) to determine the best solution for the next generation. In this step non-dominated and crowding distance procedure used to select best non dominated solution and distance (Ali et al., 2012).

Step-10, stopping criterion

If the number of generations has exceeded the maximum number of generations, *Gmax*, then the iteration stops and displays the results of the non-dominates solutions set obtained at *Gmax* iteration. Otherwise, if the generation has not reached *gmax* yet, then go back to step-4, update iteration.

3.3. DE-hybrid Algorithm

The procedure of DE-hybrid is generally same as DE-ns. Some additional procedures were inserted into the algorithm. The additional procedures included as follow:

Step 2, initialization

DE-hybrid used 3 different initialization of population, DE random generate initialization, Opposition Based Learning (OBL) initialization, and bottleneck heuristic initialization.

- Opposition Based Learning

  The concept of Opposition-Based Learning (OBL) was introduced by Tizhoosh (Tizhoosh, 2005). The main idea behind OBL is the simultaneous consideration of an estimate and its corresponding opposite estimate (i.e. guess and opposite guess) in order to achieve a better approximation of the current candidate solution. Opposite population generate by solution from random generated X_i. $X' = (X'_1, X'_2,...., X'_n)$ and calculate $X'_i = Xmin_i + Xmax_i - X_i$. For example, random population Xi = [0.24 0.56 0.31 0.97 0.66 0.18], for X_{i1} = 0.24 the opposite solution is X'_{i1} = 0 + 1 - 0.24 = 0.76. Convert X' being an individual job permutation using smallest parameter value (SPV) rule.

Table 2: Solution representation of OBL

Dimension	1	2	3	4	5	6
X_{ij}	0.24	0.56	0.31	0.97	0.66	0.18
X'_{ij}	0.76	0.44	0.69	0.03	0.34	0.82
φ_{ij}	5	3	4	1	2	6
π_{ij}	4	5	2	3	1	6

- Bottleneck Heuristic Algorithm

The bottleneck heuristic procedure for two-objective has not been applied before, we proposed addoption and hibridation of TOC based heuristic for makespan criteria(Paternina-Arboleda et al., 2008) and bottleneck based heuristic for total tardiness criteria (Chen and Chen, 2009).

- Bottleneck heuristic for minimum **makespan** criteria :
  Step 1: Identifying the bottleneck stage based on the stage with the biggest mean flow, mean flow can be calculated by the amount of processing time for all jobs at each stage divided by the number of machines that are assigned to each stage.
  Step 2: Calculate Release Time (R), the amount of processing time at all stages before the bottleneck stage for each job
  Step 3: Calculate Delivery Time (D), the difference between the total mean flow time with the amount of processing time at all stages after a bottleneck stage for each job.
  Step 4: Determine the sequence of jobs at the bottleneck stage
  Determine the sequence of jobs based on R, D and P (processing time) from the smallest value to the largest. Schedule jobs on machines based on precedent ranking. If there is more than one machine available at time t, assign the next job on the machine with the smallest workload after time t. Calculate the starting and completion times for each job on bottleneck stage
- Bottleneck heuristik for minimum **total tardiness** criteria :
  Step 1: Identifying the bottleneck stage based on the stage with the biggest mean flow
  Step 2: Calculate *Arrival time* for each job *j* at bottleneck stage *b* (AR_{bj}) and determine available time each meachine *l* at bottleneck stage *b* (AV_{lb}) where Arrival time = Release time *(R)*
  Step 3: Calculate *ready time (RT)* for each pair job *j* and machine *l*.
  $$RT_{lbj}=max\{AR_{bj},AV_{lb}\} \quad (11)$$
  Step 4: Calculate *Completion Time* for each pair job and machine *(j,l)* at bottleneck stage *b*.
  $$C_{lbj} = RT_{lbj} + p_{lbj} \quad (12)$$
  Step 5: For each C_{lbj} determine machine selection rules for calculate completion time at the last stage.
  Step 6: $Z_{lj} = max\{d_j - (C_{lij}-C_{lbj}), C_{lij}\}$ (13)
  Step 7: Determine the schedule according to *smallest value* Z_{lj}, if more than one job have same value use smallest value of *dj*, and if more than one job has the smallest value of *dj* use *smallest processing time* (p_{lbj})

- Combine Solution
  Illustration of bottleneck heuristic combination
  π^1= [1 7 6 5 4 2 3] ➔ $\varphi_j(\pi^1)$ = [1 6 7 5 4 3 2] makespan solution
  π^2 = [7 5 1 6 4 2 3] ➔ $\varphi_j(\pi^2)$ = [3 6 7 5 2 4 1] total tardiness solution
  random generated $w_1 = 0.5$ and $w_2 = 0.5$.
  $\varphi_j = w_1 * (\varphi_j(\pi^1)) + w_2 * (\varphi_j(\pi^2))$
  φ_j = 0.5 * [1 6 7 5 4 3 2] + 0.5 * [3 6 7 5 2 4 1]
  φ_j = [0.5 3 3.5 2.5 2 1.5 1] + [1.5 3 3.5 2.5 1 2 0.5] = [2 6 7 5 3 3.5 1.5]

φ_j	2	6	7	5	3	3.5	1.5
Smallest value	2	6	7	5	3	4	1
New permutation (π)	7	1	5	6	4	2	3

  New permutation = [7 1 5 6 4 2 3]

Step 2a Selection of initial solution
selection between random generate population, OBL population and bottleneck heuristic population to determine the best solution for the next generation. In this step non-dominated and crowding distance procedure used to select best non-dominated solution and distance.

4. Experiment Result and Comparison

4.1 Scheme of Experiment Data

This research addopt data scheme from carlier and Neron's benchmark problem (Carlier and Néron, 2000), there are 48 different problems and three characteristics that define a problem are the number of jobs, the number of stages, and the number of machines at each stage. For example, the notation of *j10s10a1* means a 10-job , 10-stage with *a* structure of machine layout at the stage and 1is the problem index for specific type. The problem sizes vary from 5

jobs×5 stages to 15 jobs×10 stages. The processing time of the job generated uniform distribution at range (3,20) and the machines layout are consider to 4 rules proposed by Alaykiran(Alaykýran et al., 2007):

a. There is one machine at the middle stage (bottleneck) and three machines at the other stages
b. There is one machine at the first stage (bottleneck) and three machines at the other stages
c. There are two machines at the middle stage (bottleneck) and three machines at the other stages
d. There are three machines at each stage (no bottleneck)

Table 3: Data set configuration

Factor			
Number of job, n	5	10	15
Number of stage m	5	10	
Number of machines for each stage, k	4 machine layout (a,b,c,d)		
Processing time job j at stage i, p_{ij}	U(3,20)		
Processing time job j at machine l for each stage i, p_{lij}	U(p_{ij}-2,p_{ij}+2)		
Job due date, d_j	U(μ∗m , μ∗(n+m-1)		

Total number of data set is 48 problem consisting 24 simple problem and 24 hard problem. The problem were tested by several level of parameter to find the best parameter setting for DE algorithm. The experiment result based on makespan and total tardiness. The level of each parameter, F = (0,3; 0,5), Cr = (0,5;0,7;0,9) and NP = (50,100,200). There are 36 parameter combination run for 10 replication. The obtained best parameter is F=0,3 Cr=0,9 dan NP=100.

4.2 Preliminary Experiment

The proposed algorithm was tested with the simple flow shop problems using data from OR-Library. The data Car06 (8 jobs 9 stages), Car07 (7 jobs 7 stages) and Car08 (8 jobs 8 stages) is used to determine performance of the proposed algorithm to produces a set of non-dominated solutions. The result of preliminary experiment DE hybrid algorithm gives a good performance to get a solution with convergence metric = 0, it means that the algorithm can reach the optimal PF solution. The following chart was non-dominated solutions obtained by DE-hybrid for Car08 problem.

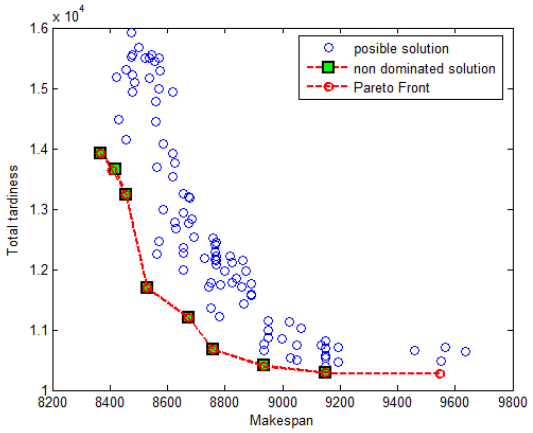

Figure 1: Non-dominated solution of problem Car08 with DE-hybrid algorithm

4.3 Experiment Result

This experiment is divided into two types based on problems, that is experiments for simple problem and experiment for hard problems. Proposed algorithm DE-ns and DE-hybrid compared to the DE algorithm based on performance metric. There are 3 performance criteria used, i.e. number of non-dominated solution (NNDS), convergence metric and divergence metric. The proposed algorithm was programmed in Matlab 7.0.1 and run on a PC with Intel Core 2 Duo CPU E4700 2.60GHz and 1024MB RAM. The experiment limited by maximum iteration parameter (Gmax) and run with 10 replication. The performance of all compared algorithms for simple problem is summarized in Table4. For simple problem DE-hybrid obtained best solution quantity with higher average NNDS than other

algorithm and good divergence metric with minimum divergence value. DE-hybrid also obtained best convergence metric, the algorithm can reach the optimal PF 23 of 24 problem (96%) with smallest average convergence value. DE-ns algorithm can reach 18 optimal PF of 24 data set problem (75%), and DE algorithm just reach 13 optimal PF (54%). But it worst in computational time than DE algorithm.

Table 4: Comparison result on simple problem with PF solution

Problem	NNDS			Convergence Metric			Divergence Metric			Computation time		
	DE	DE-ns	DE-hybrid	DE	DE-ns	DE-hybrid	DE	DE-ns	DE-hybrid	DE	DE-ns	DE-hybrid
j5s5a1	2.4	11.3	11.6	0.075	0	0	0.981	1	1	2.418	3.9156	4.5084
j5s5a2	1	11.7	11.4	0	0	0	1	1	1	2.402	4.0092	4.4772
j5s5a3	1	7	7.3	0	0	0	1	1	1	2.418	3.9624	4.4148
j5s5a4	1.9	3.9	3.9	0.467	0	0	1	1	1	2.34	3.978	4.4772
j5s5a5	1.9	8.4	7.9	0	0	0	1	1	1	2.324	3.9156	4.5552
j5s5a6	2.9	32.2	32.2	0	0	0	0.75	0.8251	0.8251	2.387	4.3992	4.8672
j5s5b1	2.1	7.1	6.9	0.447	0.285	0	0.947	0.6208	0.5274	2.246	3.8688	4.5708
j5s5b2	4.3	6.1	5.9	0.312	0.0655	0	0.701	0.7002	0.6973	2.246	3.8532	4.3368
j5s5b3	3.3	11.5	11.6	0	0	0	0.866	0.6426	0.5532	2.262	3.8064	4.3212
j5s5b4	1.4	2.1	2	0.308	0.1374	0	1	1	1	2.278	3.8064	4.3524
j5s5b5	3.7	17	18	0.215	0.0266	0.0258	0.833	0.5078	0.5131	2.231	4.1808	4.6956
j5s5b6	2.5	2	2	1.407	0.9	0	0.873	1	1	2.168	3.9312	4.3056
j5s5c1	1	4	4	0	0	0	1	1	1	2.449	4.1184	5.0232
j5s5c2	1	11.7	11.6	0	0	0	1	1	1	2.683	4.3056	4.9764
j5s5c3	2	8.6	9.8	0.05	0	0	1	1	1	2.558	4.2744	4.8828
j5s5c4	1	6	5.9	0	0	0	1	1	1	2.496	4.1652	4.6644
j5s5c5	1.9	5	5.9	0.1	0	0	1	1	1	2.496	4.1496	4.8672
j5s5c6	1.2	6.1	6.8	0.033	0	0	1	1	1	2.527	4.1652	4.8048
j5s5d1	1	2.3	2	0	0.0447	0	1	1	1	2.621	4.3212	5.0388
j5s5d2	1	11.4	11.7	0	0	0	1	1	1	2.621	4.3212	4.8048
j5s5d3	1	1	1	0	0	0	1	1	1	2.652	4.29	5.1012
j5s5d4	1	5.7	5.2	0.1	0	0	1	1	1	2.668	4.3212	4.914
j5s5d5	1	18.5	19.3	0	0	0	1	1	1	2.683	4.3368	4.9764
j5s5d6	1	1	1	0	0	0	1	1	1	2.605	4.2744	4.8048
Mean	1.771	8.4	8.5375	0.146	0.0608	0.0011	0.956	0.929	0.9215	2.449	4.1113	4.6976

The performance of all the compared algorithm for partially hard problem is summarized in table 5. For hard problem, there are unknown optimal PF solution. Since optimal PF is not known for any of the test problem, seven points of comparison are generated for each problem to create a measure of the algorithm's efficacy (Schott, 1995).

Table 5: Comparison result on hard problem

Problem	NNDS			Convergence			Divergence		
	DE-murni	DE-ns	DE-hybrid	DE-murni	DE-ns	DE-hybrid	DE-murni	DE-ns	DE-hybrid
j10s10a	2.2	44.2	45.7	160.9304	32.6233	31.8757	0.9424	0.6726	0.6933
j10s10b	1.8	11	11	171.7701	67.1861	67.1955	1	0.9701	0.9137
j10s10c	2.2	45.7	46.4	3.86554	3.82464	3.65224	1	0.57786	0.52869
j10s10d	1.6	22.5	25.3	7.02495	4.2707	3.7661	1	0.755	0.75035
j15s10a	3.5	100.4	113.8	159.498	31.9312	28.3843	0.892	0.6398	0.6781
j15s10b	3	60.9	59.7	168.4867	37.2636	38.4873	0.8582	0.6652	0.6435
j15s10c	6.9	34.1	34.9	21.04361	16.2851	15.9094	0.69592	0.6828	0.6933
j15s10d	5.7	11.9	14.7	3.5862	4.8528	3.7073	0.8631	0.81015	0.80654
Average	3.36	41.34	43.94	87.03	24.78	24.12	0.91	0.7217	0.7134

5. Conclusion

The main contribution of the paper is to develop a new approach hybridation DE with bottleneck heuristic to exploit the bottleneck stage and to make better inisialization. In addition, nondominated sorting and crowding distance proposed to change selection mechanishm of DE algorithm. Non-dominated sorting used to find simultaneous solution of two objective HFS-UPM problem. To evaluate the algorithm, several problem from carlier and neron are used to compare proposed algorithm DE-ns and DE-hybrid. By comparing with DE for HFS-UPM, the proposed algorithm found better solution. All simulation results and comparisons demonstrated the efectiveness of the proposed DE-hybrid algorithm. The future work is to apply the DE-hybrid algorithm to the problem with another objectives, and we will develop multi-objective DE algorithm for other kinds of combinatorial optimization problems.

References

Alaykýran, K., Engin, O. & Döyen, A. 2007. Using ant colony optimization to solve hybrid flow shop scheduling problems. *The International Journal of Advanced Manufacturing Technology,* 35**,** 541-550.

Ali, M., Siarry, P. & Pant, M. 2012. An efficient Differential Evolution based algorithm for solving multi-objective optimization problems. *European Journal of Operational Research,* 217**,** 404-416.

Carlier, J. & Néron, E. 2000. An exact method for solving the multi-processor flow-shop. *RAIRO-Operations Research,* 34**,** 1-25.

Chen, C.-L. & Chen, C.-L. 2009. Bottleneck-based heuristics to minimize total tardiness for the flexible flow line with unrelated parallel machines. *Computers & Industrial Engineering,* 56**,** 1393-1401.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P. & Werner, F. 2009. A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *Computers & Operations Research,* 36**,** 358-378.

Li, X. & Yin, M. 2013. An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure. *Advances in Engineering Software,* 55**,** 10-31.

Onwubolu, G. & Davendra, D. 2006. Scheduling flow shops using differential evolution algorithm. *European Journal of Operational Research,* 171**,** 674-692.

Paternina-Arboleda, C., Montoya-Torres, J., Acero-Dominguez, M. & Herrera-Hernandez, M. 2008. Scheduling jobs on a k-stage flexible flow-shop. *Annals of Operations Research,* 164**,** 29-40.

Pinedo, M. 2008. *Scheduling: theory, algorithms, and systems*, Springer Science + Business Media.

Qian, B., Wang, L., Huang, D.-X., Wang, W.-L. & Wang, X. 2009. An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers. *Computers & Operations Research,* 36**,** 209-233.

Quadt, D. & Kuhn, H. 2007. A taxonomy of flexible flow line scheduling procedures. *European Journal of Operational Research,* 178**,** 686-698.

Ruiz, R. & Vázquez-Rodríguez, J. A. 2010. The hybrid flow shop scheduling problem. *European Journal of Operational Research,* 205**,** 1-18.

Schott, J. 1995. *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization.* Massachusetts Institute of Technology.

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**,** 1930-1947.

Tizhoosh, H. R. Opposition-Based Learning: A New Scheme for Machine Intelligence. Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on, 28-30 Nov. 2005 2005. 695-701.

Vesterstrom, J. & Thomsen, R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. Evolutionary Computation, 2004. CEC2004. Congress on, 19-23 June 2004 2004. 1980-1987 Vol.2.

Biography

Riza Auliya Rahman is a lecturer at Industrial Engineering Department, Universitas Brawijaya (UB), Malang, Indonesia. He earned B.S. (S.T.) in Industrial Engineering from UB, Malang, Indonesia. Masters in Industrial Engineering from Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia. He is currently an assistant of

Industrial Computation and Optimization Laboratory ITS. His research interests include optimization, simulation and scheduling.

Budi Santosa is a professor and Head of Department at Industrial Engineering Department, Institut Teknologi Sepuluh Nopember (ITS), Surabaya Indonesia. He earned B.S. in Industrial Engineering from Institut Teknologi Bandung (ITB), Indonesia, Masters and PhD in Industrial Engineering from University of Oklahoma, USA. He has published journal and conference papers. Dr Budi Santosa has done research projects in application of methaeuristics techniques in logistics, transportation and scheduling. His research interests include optimization, metaheuristics, scheduling and data mining.

Stefanus Eko Wiratno is a lecturer at Industrial Engineering Department, Institut Teknologi Sepuluh Nopember (ITS), Surabaya Indonesia. He earned B.S. in Industrial Engineering from Institut Teknologi Sepuluh Nopember (ITS), Indonesia, Masters in Industrial Engineering from Institut Teknologi Bandung (ITB), Indonesia. He has published journal and conference papers. His research interests include simulation, scheduling and system modeling.