IJE TRANSACTIONS C: Aspects Vol. 27, No. 9, (September 2014) 1395-1404

International Journal of Engineering

Journal Homepage: www.ije.ir

Multi-objective Differential Evolution for the Flow Shop Scheduling Problem with a Modified Learning Effect

H. Amirian*, R. Sahraeian

Department of Industrial Engineering, College of Engineering, Shahed University, Tehran, Iran

PAPER INFO

Paper history:
Received 26 January 2014
Received in revised form 31 March 2014
Accepted 22 May 2014

Keywords:
Differential Evolution
Multi-Objective Scheduling
Flow shop
Truncated Dejong's Learning Effect

ABSTRACT

This paper proposes an effective multi-objective differential evolution algorithm (MDES) to solve a permutation flow shop scheduling problem (PFSSP) with the modified Dejong's learning effect. The proposed algorithm combines the basic differential evolution (DE) with local search and borrows the selection operator from NSGA-II to improve the general performance. First the problem is encoded with an appropriate rule to make the continuous nature of DE suitable for flow shop problems. Second, insert based local search is added in the initialization stage, as well as in each iteration to speed up convergence. The former guarantees that the algorithm commences with better solutions while the latter focuses the algorithm on promising areas. Third, in each generation, in order to improve diversity, two populations are introduced, current pop and advanced pop. The best solutions of each iteration are stored in the current pop, while the less than desirable solutions are added to the advanced pop. At the end of each generation, the two are combined and better individuals are selected for the next generation. The algorithm is then tested on benchmark problems to demonstrate its effectiveness and the results are discussed. Finally, a truncated version of Dejong's learning effect is proposed and MDES is used to solve the permutation flow shop with the modified learning effect.

doi: 10.5829/idosi.ije.2014.27.09c.09

1. INTRODUCTION

Scheduling is the essential element of survival in the marketplace for any production, manufacturing system and service industry. Consequently, it is necessary to develop fast, efficient and practical approaches in scheduling [1]. Among the different environments of scheduling, flow shops are of the most important and well-known problems, and have been proved to be strongly NP-hard even when only two machines are considered [2]. Unfortunately, most proposed models are only effective in theory since for the sake of simplicity some practical assumptions, such as release dates, machine breakdowns, blocking, and setup times are usually ignored. Hence, the credibility of the model in the real world is lost. Learning effect is one of such practical assumptions. Learning effect states that the production facility performance is improved continuously with time. As a result, the processing time of a given job is shorter if it is scheduled later, rather than earlier in the sequence. This phenomenon is known as the ‘‘learning effect’’ in the literature [3]. Adding learning effect considerations to an already NP-hard *m* machine permutation flow shop problem and considering multiple objectives simultaneously, enhance the complexity of the problem. Thus, it is necessary to develop effective and efficient approaches for such a problem. The concept of learning effect was first introduced in scheduling by Biskup [4]. Many studies have since been conducted in this area of scheduling. According to an extensive review by Biskup [5], learning effect can be categorized into two main classes: position-based and the sum-of-processing-time approaches. The first focuses on the position of each job and is directly affected by the number of jobs in the process. The second adds up the processing time of all

*Coresponding Author’s Email: amirian012univ@ymail.com(H. Amirian)

Please cite this article as:H. Amirian and R. Sahraeian, Multi-objective Differential Evolution for the Flow shop Scheduling Problem with a Modified Learning Effect,International Journal of Engineering (IJE), TRANSACTIONS C: Aspects Vol. 27, No. 9, (September 2014) 1395-1404

jobs processed so far. Most studies, however, concentrate on the classic position-based learning effects. Eren & Guner [3], used the classic version in a bi-criteria flow shop scheduling environment. They considered the minimization of weighted sum of total completion time and make-span with only two machines. Chung & Tong [6] used the same classic learning effect to solve an *m*-machine flow shop scheduling problem reliably up to 18 jobs. Biskup [5] brought up the question that among different types of learning effects; which type should be used to represent the reality best. The answer depends on the production environment. Thus, many studies tried to modify the position-based version to show the reality of learning better. Okolowski & Gawiejnowicz [7], modified the formulation to make it suitable for both machine based and operator based jobs in parallel machine environments. They introduced a parameter (*M)* to represent the part of job processing time that cannot be shortened due to some restrictions e.g., fixed machine times. Similarly, Cheng et al. [8], tried to improve the classic version by introducing a truncation parameter. According to them, under the classic learning model, the actual processing time of a job drops to zero precipitously as the number of jobs increases, which is at odds with reality. Their model prevents such occurrence by proposing lower bounds on the processing times i.e., the actual processing time of a job cannot be lower than its normal processing time multiplied by truncation parameter. The real-world problems usually involve the optimization of several objectives simultaneously. Since the late 1980s, many multi-objective problems are confronted in manufacturing systems [9] and thus it brings up the need to study scheduling in multi-objective environments. According to an extensive review by Sun et al. [10] on the multi-objective flow shop optimization algorithms, there are basically two approaches to solve a multi-objective flow shop problem, exact and approximation methods. Among many exact methods, mostly branch-and-bound has proven useful in tackling small-sized problems [7, 8, 11]. However, since this method is still incapable of solving medium and large instances (too much computational time), approximation methods, especially meta-heuristics have been developed in recent years as attractive alternatives. These efficient meta-heuristic methods mainly include genetic algorithms (GA), particle swarm optimization (PSO), ant colony optimization (ACO), simulated annealing (SA), tabu search (TS), and differential evolution (DE) [10]. Among these methods, GA as an evolutionary algorithm has attracted lots of interest for solving large-sized problems. Cheng et al. [2] proposed an adaptive genetic local search algorithm for PFSSP to minimize make-span and total flow time simultaneously. They used a dynamic population size and a local search method to improve their algorithm. Their selection scheme is a hybridization of PESA-II [12] and NSGA-II [13]. It combines the advantages of both algorithms to enhance the diversity of NSGA-II and decrease the too strong selective pressure of PESA-II. They show that their adaptive method (MPFA) outperforms similar algorithms i.e., PASA [14] for small-sized problems and MOSA [15] and PGA-ALS [16] for large-sized problems. Another common technique is the use of evolutionary algorithms to solve scheduling problems. As a relatively new evolutionary technique, differential evolution has gained much attention due to its simple implementation, robustness and quick convergence. Like other evolutionary algorithms, it includes three main operators; mutation, crossover and selection. However, due to its continuous nature, DE is rarely used in scheduling problems. Qian et al. [17], first used DE in multi-objective flow shop environment. They proposed a hybrid DE (MOHDE) to minimize a bi-objective problem with the objectives being the make-span and maximum tardiness. In using DE for discrete problems such as flow shop scheduling, we usually need to add another search method to improve the performance of the algorithm. Thus, they also used a method named variable neighborhood search (VNS) incorporated within their algorithm. In their next paper, Qian et al. [18] added a no-wait condition to the PFSSP problem and solved it with a memetic algorithm based on differential evolution (MADE). They introduced a very simple, but useful encoding scheme known as the largest-order-value rule (LOV) to convert continuous values of DE to job permutations. Later, Qian et al. [9], solved a PFSSP with limited buffers by a hybrid DE (HDE). They used a DE/rand-to-best/1/exp scheme and an insert-based local search to tackle the mentioned problem. The local search was carried out on 1/5 individuals in each iteration.

Similarly, we use multi-objective DE for permutation flow shop scheduling problem with a learning effect. To the best of our knowledge, no other work in solving PFSSP with DE has considered learning effects. First, we use the LOV rule [9] to make DE suitable for PFSSP. Second, we use an insert-based local search in the initialization step so the algorithm starts with better population. Third, in addition to insertion operator in each generation, we borrow the NSGA-II selection operator, based on crowding distance and non-dominated sorting, to select the individuals for the next generation. Fourth, the less than desirable solutions of each iteration are not discarded, but are added to another population called advanced population. This new population seeks to enhance diversity [19]. Finally, after evaluating the algorithm using MO-PFSSP test suits, we solve the PFSSP with a truncated version of Dejong's learning effect. The rest of this paper is organized as follows. In section 2, the mathematical model of the modified learning effect and the objectives are described. Section 3 gives a brief introduction to DE. In

section 4, the proposed algorithm, namely MDES, is explained extensively by the pseudo code and flowchart of the MDES. In section 5, computational experiments, benchmark results and performance metrics are discussed. The paper is then concluded in section 6.

2. MATHEMATICAL MODEL

Le *m* denote the number of machines, *i=1,2,…,m.* Now in a flow shop problem *n* jobs, *j=1,2,…,n,* must go through all the *m* machines. If we assume the same sequence for all the machines, we will have permutation flow shop scheduling problem (PFSSP).

2. 1. Objective Functions Given the job permutation $\pi=\{\pi(1),\pi(2),\ldots,\pi(n)\}$ where $\pi(j)$ denotes the job J which is in the position j of π, and $p_{i,j}$ is the normal processing time of job j on machine i, the completion time i.e., make-span C_{max} is formulated as follows [2]:

$$C_{max} = \max_j\{C_{m,p(j)}\}. \quad (1)$$

The total flow time (TFT) can also be calculated using the following formula:

$$TFT = \sum_{j=1}^{n} C_{m,p(j)}. \quad (2)$$

2. 2. Learning Effect Formulation Cheng et al. [8] introduced a truncation parameter, b $(0 < b < 1)$, as a default limit that prevents the irrational decrease in processing times. Hence, $p_{i,j,r}$, the actual processing time of job *j* on machine *i* in position *r; (r=1,…,n),* is formulated as follows:

$$p_{i,j,r} = p_{i,j}.\max\{r^a, b\}, \quad (3)$$

where *a* is the learning effect parameter, *a<0*. This type of learning model is designed for manual jobs where the operator is gaining experience as the time goes by. However, nowadays most of the operations carried out on a job are a result of human-machine interactions. Generally, the machine-based part of the processing time of a job, is fixed and cannot be shortened. Okolowski & Gawiejnowicz [7] considered this issue and proposed the Dejong's learning effect with an incompressibility factor *M*. Their model is formulated as follows:

$$p_{i,j,r} = p_{i,j}(M + (1-M).r^a). \quad (4)$$

Different values of *M* are suggested in the literature. For example, *M=0.25* is usually used for labor-intensive jobs and *M=0.5* for machine-intensive jobs [7]. We propose a truncated version of Dejong's approach with the following formula:

$$p_{i,j,r} = p_{i,j}(M + (1-M).\max\{r^a, b\}). \quad (5)$$

This modified version carries the advantages of both methods, i.e., it prevents the processing times from falling to zero when *M=0*, and considers the fixed times (e.g., machine-based times) in the process at the same time.

3. BRIEF INTRODUCTION TO DIFFERENTIAL EVOLUTION (DE)

Differential evolution [20] is basically a population based evolutionary algorithm which was first introduced in continuous space to optimize real parameters and real valued functions. The initialization stage in DE starts with a random population of NP solutions (target vector): $x_{i,G}; i = 1,\ldots, NP$ where i denotes the ith individual in the population and G indicates the current generation. Then the algorithm enters the main loop where each individual undergoes mutation, crossover and selection.
Mutation: this operator expands the search space and is the core of DE. For a given vector $x_{i,G}$ randomly select three vectors $x_{r1,G}, x_{r2,G}$ and $x_{r3,G}$ such that the indices *i*, *r*1, *r*2 and *r*3 are distinct. Calculate the donor vector $V_{i,G}$ by adding the weighted difference of two of the vectors to the third vector known as base vector [19]:

$$V_{i,G} = x_{r1,G} + F \times (x_{r2,G} - x_{r3,G}), \quad (6)$$

where F is a constant known as the mutation factor or control parameter, F ϵ [0,1]. The value of F determines how heavily the donor vector is affected by this difference.
Crossover: in the binomial crossover, *the trial vector* $U_{i,G} = (u_{1,i,G},\ldots,u_{n,i,G})$, is either developed from the elements of the target vector $X_{i,G} = (x_{1,i,G},\ldots,x_{n,i,G})$, or that of the *donor vector* $V_{i,G} = (v_{1,i,G},\ldots,v_{n,i,G})$. The choice is determined by the crossover rate CR [0, 1] and a random parameter's index jj ò $\{1,\ldots,n\}$ for *j* individuals, $j = 1,\ldots,n$:

$$u_{j,i,G} = \begin{cases} v_{j,i,G} & \text{if } rand_j \le CR \text{ or } j = jj \\ x_{j,i,G} & \text{otherwise} \end{cases} \quad (7)$$

Selection: another factor that differentiates between DE and other evolutionary algorithms is the selection scheme. In this phase, the trial and target vectors are evaluated and the one with lower objective value is selected for the next generation. Thus, the trial solution is compared against not all but one solution, its counterpart in the current generation [19].

$$X_{i,G+1} = \begin{cases} U_{i,G} & \text{if } f(U_{i,G}) \le f(X_{i,G}) \\ X_{i,G} & \text{otherwise} \end{cases} \quad (8)$$

4. MDES FOR PFSSP

4. 1. Solution Representation Due to the continuous nature of DE, a scheme is needed to convert the individuals extracted by DE to a sequence/job permutation. In the present paper, we use the largest-order-value (LOV) rule [9] to get the job solution $p_i = (p_{i,1},...,p_{i,n})$ from the original DE values X_i. According to LOV rule, $X_i = (x_{i,1},...,x_{i,n})$ vector is sorted in descending order to get a sequence $j_i = (j_{i,1},...,j_{i,n})$. The job permutation π_i is then calculated by the following formula:

$$p_{i,j_{i,j}} = j. \quad (9)$$

Since the DE operators are based on the continuous space, every time a new vector of X_i is generated, in order to calculate the objective function(s), it should be converted to a job sequence using the LOV rule. Moreover, if due to the operators of local search, such as swap, reverse or insert, the sequence is changed, X_i vector, should be repaired accordingly, i.e., its corresponding job permutation should match the permutation resulted by the local search. According to Qian, et al. [9], the repair process has two steps:

Step 1: Calculate the sequence j_i using the following formula:

$$j_{i,p_{i,j}} = j. \quad (10)$$

Step 2: Values of the vector X_i, are sorted according to the new vector j_i.

4. 2. MDES Based on NSGA-II In this section, the procedure of the proposed algorithm is discussed in detail.

4. 2. 1. Initialization with Insert based Local Search Naturally, feeding better solutions to the algorithm in the initialization phase improves the general performance of the algorithm. To this end we have used an insert based local search in this phase. Insertion operator is widely used for flow shops and is known to give good results in practice. Insertion operator deletes an element situated at a position i and reintroduces it in another position *j; i,j= {1,..,n}*. The jobs situated between positions i and j are therefore shifted [9].

4. 2. 2. Mutation & Crossover In the proposed algorithm, we use the concept of random localization [19] to choose the base vector in mutation. According to this rule after selecting three distinct solutions randomly, $X_{r1,G}$, $X_{r2,G}$ and $X_{r3,G}$ from the population for the target solution $X_{i,G}$, a tournament is then held among the three solutions and the solution with the best fit is chosen as the base vector $X_{Best,G}$. The mutation formula then becomes:

$$V_{i,G} = X_{Best,G} + F \times (X_{r2,G} - X_{r3,G}). \quad (11)$$

This method seeks to find better solutions by improving the vector which effects donor vector the most, i.e. *the base vector*. Additionally, the use of random localization in mutation prevents the search from becoming a purely random or a purely greedy search. After completing the mutation phase, MDES performs crossover, as defined by Equation (7), and enters the selection phase [19].

4. 2. 3. NSGA-II Based Selection In the proposed algorithm, similar to Ali et al. [19], we use two-selection operator. First, in the inner loop each trial solution is compared with its target counterpart according to Equation (8). The winner of the competition enters the current population, while the other is selected for the advanced population. Second, we also borrow the selection operator from NSGA-II [13]. At the end of each iteration, the current and advanced populations are combined. Rank and crowding distance are calculated for each individuals. The truncated population for next generation are solutions with lower rank and greater crowding distance, respectively. This guarantees that the next generation is as good as, or better than the current generation; hence the name evolutionary [21-24].

4. 2. 4. Local Search with Insertion Operator At the end of each generation, a new neighbor is generated for each NP solutions by insertion operator. This new solution is then evaluated and compared with its counterpart both in current and advanced population. If either of the two target solutions are dominated by the new neighbor, they are replaced by it in their corresponding population. The idea stems from the need to maintain balance between diversity and convergence in any evolutionary algorithm. Finding a near neighbor for each solution helps speeding up the convergence since it concentrates on a specific region and give it a thorough search. On the other hand, if the neighbor turns out to be less desirable than the original solution, instead of discarding it, we add it to the advanced population so that it would have a chance to compete with other solutions; thus enhancing the diversity.

4. 3. Procedure of MDES The procedure of MDES is summarized as follows:

Step 1- Problem definition: Set initial population size ($Npop$), number of objectives ($Nobj$), crossover rate (*CR*), mutation control parameter (*F*), initial upper bound ($Var\max$) and lower bound ($Var\min$), maximum number of iterations ($G = 1,..., \max it$) and load scheduling model (*n* jobs; *(j=1,…,n),* *m* machines*(i=1,…,m)* and processing times).

Step 2- Initialization Phase:

2.1. Generate NP random solutions ($X_{i,j} = j^{th}$ individual of solution vector X_i; $i = 1,..., NP$

$$X_{i,j} = \text{varmin}(X_{i,j}) + rnd(0,1) * (\text{varmax}(X_{i,j}) - \text{varmin}(X_{i,j})) \quad (12)$$

2.2. Convert $X_{i,j}$ to p_i according to LOV rule

2.3. Evaluate the job permutations p_i

2.4. Insert Based Local Search: p_{i_new}= *Insertion*(p_i): Randomly select two jobs *i, j*:

if $i < j$: $p_{i\_new} = [p_i(1:i-1), p_i(i+1, j), p_i(i), p_i(j+1:end)]$ Else

$p_{i\_new} = [p_i(1:j), p_i(i), p_i(j+1:i-1), p_i(i+1:end)]$

if p_{i_new} *dominates* p_i : $p_i = p_{i\_new}$

2.5. Convert p_i to X_i using repair process

Set G=1;

Step 3- Main Loop:

3.1. Insert Based Local Search

3.2. Sort population (rank and crowding distance)

3.3. Set *Current pop = pop*

3.4. Inner Loop: *Repeate for all NP individuals*

3.4.1. Mutation & Crossover Phase (Equation (7), (11)).

3.4.2. If $U_{i,G}$ dominates $X_{i,G}$: Replace $X_{i,G}$ with $U_{i,G}$ in the *current pop* &Add $X_{i,G}$ to the *advanced pop*

Else: Add $U_{i,G}$ to *advanced pop*

*3.4.3.*Insert Based Local Search: p_{i_new}= *Insertion* (p_i),

If p_{i_new} dominates current pop (p_i) :*Current pop*(p_i) = p_{i_new} &*Tempt pop = Current pop*(p_i)

Elseif p_{i_new} dominates *advanced pop* (p_i): *Advanced pop* (p_i) = p_{i_new}

3.4.4. NSGA-II Selection (Combine and Truncate):

Pop = [Current pop; Advanced pop; Tempt pop]

Non-dom_Sort_Crowding_Distance (pop)

G=G+1; if *G<max_it repeat step 3*

Additionally, the framework of MDES is illustrated in Figure 1. As it can be seen using the additional populations (advanced and tempt pop) has improved exploitation. It has to be noted that the nature of advanced and tempt pop are the same, so only one of them is discussed in the paper.

5. COMPUTATIONAL EXPERIMENTS

We use 7 well-studied benchmarks for multi-objective scheduling (i.e. Ta001, Ta002, Ta003, Ta004, Ta005, Ta051, Ta057) to evaluate the proposed algorithm without learning effect considerations. Once we are certain of the effectiveness of thealgorithm, we test it on the PFSSP with truncated Dejong's learning effect.

5. 1. Experimental Setup In this paper by try-and-error and according to the results of 10 runs of 230 generation for each problem, DE parameters are set as *CR=0.2* and *F* [0.6023,0.8023]. Let *Npop* =50, max_it=100, $Var\min$ =1, $Var\max$ =10, $Nobj$ =2.

5. 2. Performance Metrics Here we use four performance metrics to test reliability, efficiency, and robustness of our proposed MDES.

5. 2. 1. Overall Non-dominated Solution Number (ONSN) Consider S the set of desired non-dominated solutions. Let s_j be the set of non-dominated solutions obtained from the algorithm under test. ONSN counts the number of those solutions in s_j not dominated by any other solution of set S. Consequently, a measure of good performance of s_j is the higher count of its ONSN [9].

5. 2. 2. Overall Non-dominated Vector Generation (ONVG) In this metric, the number of non-dominated solutions found by the obtained set s_j is counted. The idea behind this is that the algorithm with higher ONVG has explored the solution space more thoroughly, hence it is better than its counterpart. Simply it is defined as $|s_j|$ [9].

5. 2. 3. Diversity Metric **(Δ)** Deb et al. [13] introduced a diversity metric to gauge the extent of spread achieved among the obtained solutions. Assuming that there are *N* solutions on the best-non-dominated front, this metric is given by:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}}, \quad (13)$$

where $d_i; (i = 1,..., N-1)$ is the Euclidean distance between consecutive solutions in the obtained non-dominated set and $\bar{d}$ is the average of all d_i. Here d_f and d_l refer to the Euclidean distances between the extreme solutions and the boundary solutions of the

obtained non-dominated set, respectively. The smaller Δ is more desirable.

5. 2. 4. Average Quality (AQ) Qian et al. [9] proposed the modified version of AQ to measure the quality of the solution set. The modification was applied to prevent the original AQ from hiding some quality aspects of the solutions in terms of diversity and convergence. They added diversity indicators to the model and formulated the metric as follows:

$$AQ = \sum_{l \in \Lambda} \frac{s_a(f, z^0, l, r)}{|\Lambda|}, \quad (14)$$

where

$$s_a(f, z^0, l, r) = \min_i \{\max_j \{l_j (f_j(x_j) - z_j^0)\} + r \sum_{j=1}^{w} l_j (f_j(x_j) - z_j^0) \quad (15)$$

and $f_j(.)$is the j^{th} objective in the ω-dimensional objective space ; $i=1,\ldots,NP$, $j=1,\ldots,\omega$. In the formulation l_j is the weight assigned to each objective such that:

$$\Lambda = \{ l = (l_1,..., l_w) \mid l_j \in \{0, \frac{1}{r}, \frac{2}{r}, ..., 1\}, \sum_{j=1}^{w} l_j = 1\}. \quad (16)$$

Similar to Qian et al., we set $z^0 = (0,0)$ as the reference point in the objective space, $\rho=0.01$ and $r=100$. A smaller AQ represents better solutions.

5. 3. Comparison of MDES with PASA/MPFA/MOSA/PGA-ALS The results on different test problems are compared to those of other algorithms. According to the available benchmarks, for small-sized problems (number of jobs * number of machines), our algorithm is compared with PASA [14] and MPFA [2] for 20×5 problem size. For large sized problems (50×20), the proposed MDES is compared with MOSA [15], PGA-ALS [16], and MPFA. Since MPFA outperforms PASA, PGA-ALS and MOSA in all problems, we only need to test the effectiveness of our algorithm over MPFA. As can be seen in Table 1[1] , in the test problems Ta001, Ta003, Ta004 and Ta005, the proposed MDES is superior to other methods in terms of diversity and convergence since they have higher ONSN and ONVG and smaller Δ and AQ. For test problems Ta002 and Ta051, the diversity metrics are higher than their counterparts. This shows that the solutions are not as well spread as other methods. For test problem Ta057, an interesting argument can be given. As can be seen in Table 1, the number of ONSN and ONVG are lower than other methods. However, only five out of 20 non-dominated points achieved are dominated by other algorithms. On the other hand, from 59 points found by other methods, 40 of them are dominated by our proposed MDES. Hence, if a decision maker (DM) is looking for a single option, our points offer a better set than other methods e.g. MPFA since the number of options are limited, then their quality are enhanced. In the following figure, the Pareto front of problem TA057 yielded by our method is compared with that of MPFA (Figure 2). The performance of the algorithm in different generations (*G*) on test problem TA057 is shown in Figure 3. According to the figure, as the algorithm continues, better convergence, diversity and more non-dominated points are achieved.

5. 4. Test on Effectiveness of Insert Based Local Search in Initialization In this section, to see the effectiveness of insert operator in the initialization phase, we test the algorithm on three of Talliard's problems; first with and then without the local search.

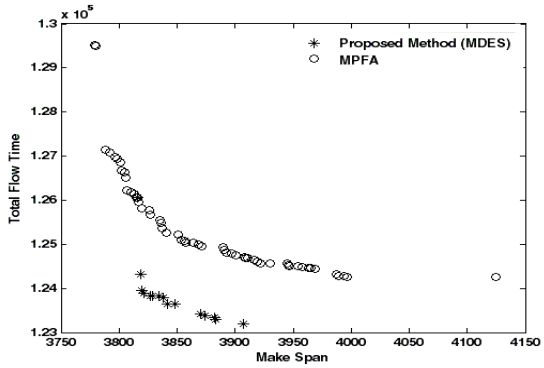

Figure 2. Comparison of the fronts achieved by the MDES and MPFA on TA057.

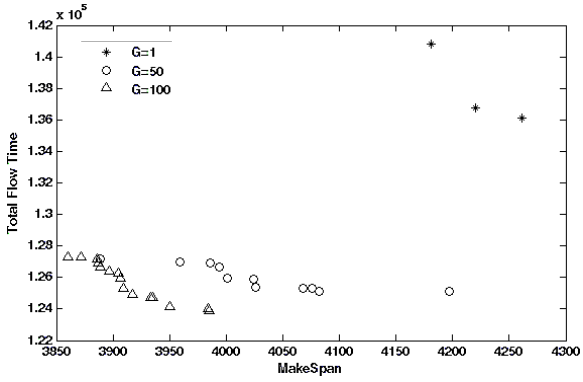

Figure 3. Convergence of the MDES on TA057 for 100 generations.

[1]The results are achieved from 10 trials of the algorithm.

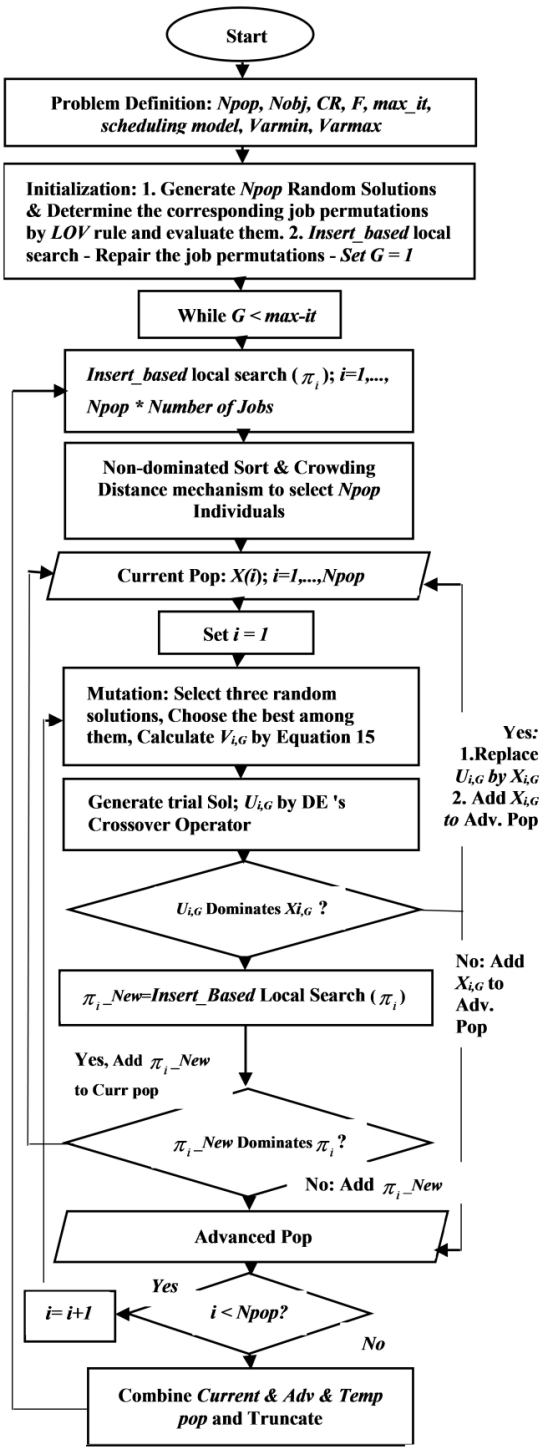

Figure 1. Framework of MDES

It can be summarized from Table 2 that the local search has improved the proposed algorithm i.e., the first two performance metrics have higher count indicating better convergence and lower Δ which shows better diversity. A smaller AQ is also achieved when using local search which indicates a better quality of the solutions.

5. 5. Test on the DE/rand-to-best/1/exp In their proposed HDE, Qian et al. [9] use the DE/rand-to-best/1/exp scheme to perform parallel exploration for flow shops with limited buffers. Inspired by this application, we tested this scheme on the proposed algorithm. The results however, show that DE/best/1/bin is more suited for our algorithm (Table 3). It can be concluded from Table (3) that the DE/best/1/bin dominates DE/rand-to-best/1/exp in all the test problems. The scheme results in higher non-dominated points i.e. ONSN & ONVG and lower AQ & Δ which all in all represents a better solution set.

5. 6. Performance of MDES with Truncated Dejong's Learning Effect We also test the proposed MDES on problems with different sizes with learning effect. The simulation results of this test are stored on "http://le-scheduling.blogfa.com" for learning rates 90%, 80% and 70% (which corresponds to *a=-0.152, a=-0.322 and a=-0.515*).The truncation rate (β) and Dejong's parameter (M) are set as 0.25 and 0.5 respectively. Table 4 shows the comparison of objective functions of the first member of the final Pareto front yielded in both cases; objective functions of the first member of the final Pareto front yielded in both cases; i.e., with and without learning effect. As expected the values yielded for each objective function are lower when learning effect is considered since as the time goes by the operator becomes more skilled and in turn the time for completing the operation decreases. For comparison purposes, the non-dominated front ofTA057 yielded with learning effect is shown in Figure 4.

6. CONCLUDING REMARKS

The pape has examined the *m*-machine permutation flow shop problem with a modified learning effect in a bi-objective environment with the objectives being the make- span and total completion time.

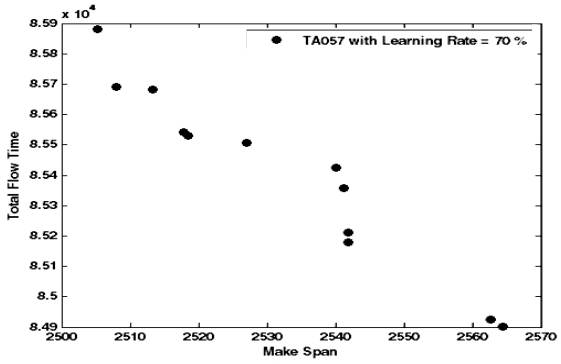

Figure 4. Pareto Front achieved by MDES on TA057(a=70%)

TABLE 1. Comparison of MDES with other algorithms

Test Problem	Job×Machine	Proposed method MDES				PASA/MPFA/MOSA/PGA-ALS			
		ONSN	ONVG	Δ	AQ	ONSN	ONVG	Δ	AQ
Ta001	20 × 5	**5.00**	**5.00**	**0.70**	**10.86**	**5.00**	**5.00**	**0.70**	**10.86**
Ta002	20 × 5	**11.00**	**11.00**	0.43	**23.48**	9.00	9.00	**0.35**	27.65
Ta003	20 × 5	**16.00**	**16.00**	**0.38**	**10.66**	**16.00**	**16.00**	**0.38**	10.76
Ta004	20 × 5	**23.00**	**23.00**	**0.55**	**12.59**	20.00	20.00	0.62	12.98
Ta005	20 × 5	**20.00**	**21.00**	**0.38**	**11.92**	**20.00**	20.00	0.40	12.21
Ta051	50 × 20	**35.00**	**35.00**	0.75	**35.10**	30.00	31.00	**0.72**	39.60
Ta057	50 × 20	15.00	20.00	**0.20**	**50.23**	**19.00**	**59.00**	0.73	101.01

TABLE 2. MDES with (out) local search in initialization

Test Problem	Job×Machine	MDES with Local Search				MDES without Local Search			
		ONSN	ONVG	Δ	AQ	ONSN	ONVG	Δ	AQ
Ta001	20 × 5	**5.00**	**5.00**	**0.70**	**10.86**	2.00	4.25	0.83	28.48
Ta003	20 × 5	**16.00**	**16.00**	**0.38**	**10.66**	5.00	9.00	0.58	12.24
Ta005	20 × 5	**20.00**	**21.00**	**0.38**	**11.92**	7.00	13.00	0.63	16.88

TABLE 3. MDES with different mutation & crossover schemes

Test Problem	Job×Machine	DE/rand-to-best/1/exp				DE/best/1/bin			
		ONSN	ONVG	Δ	AQ	ONSN	ONVG	Δ	AQ
Ta001	20 × 5	2.00	3.50	1.00	25.67	**5.00**	**5.00**	**0.70**	**10.86**
Ta003	20 × 5	3.00	7.00	0.93	11.10	**16.00**	**16.00**	**0.38**	**10.66**
Ta005	20 × 5	4.00	5.75	0.94	12.93	**20.00**	**21.00**	**0.38**	**11.92**

TABLE 4. MDES with (out) learning effect consideration

Test Problem	Job×Machine	MDES with Modified LE			MDES without LE	
		LR	C_{max}	$\sum C_j$	C_{max}	$\sum C_j$
Ta005	20 × 5	70%	812.94	9875.97	1360	13552
		80%	1004.86	10550.94		
		90%	1053.47	12609.41		
Ta057	50 × 20	70%	2564.38	84902.80	3914	123800
		80%	2749.52	92792.24		
		90%	3206.52	105834.20		
Ta071	100 × 10	70%	1693.87	98671.91	5964	309136
		80%	1968.24	119924.99		
		90%	3452.30	197065.02		

First, a truncated version of Dejong's learning effect was proposed for the flow shop problem.Then, due to the high complexity ofthe model, a hybrid differential evolution (DE) algorithm was proposed to solve the problem. The algorithm combined the non-dominated sorting and selection methods, borrowed from NSGA-II, with the classic DE and introduced an advanced population to store the less than desirable solutions of each iteration, hence improving diversity. Furthermore, an insert based local search is embedded in the initialization stage as well as the main loop of the algorithm to improve exploration. The computational results and performance metrics showed the efficiency of the proposed method. In the end since a great portion of industries are job shops; examining *m*-machine job shops with learning effect can be attractive for future research. Moreover, testing flow shops with other learning effects such as sum-of-processing-time-based learning and induced learning can be considered in the future.

7. REFERENCES

1. Fakhrzad, M., Sadeghieh, A. and Emami, L., "A new multi-objective job shop scheduling with setup times using a hybrid genetic algorithm", ***International Journal of Engineering-Transactions B: Applications***, Vol. 26, No. 2, (2012), 207-211.
2. Cheng, H.C., Chiang, T.C. and Fu, L.C., "Multiobjective permutation flow shop scheduling by an adaptive genetic local search algorithm", ***IEEE World Congress on Computational Intelligence, Hong Kong: Evolutionary Computation. CEC***, (2008), 1596 - 1602.
3. Eren, T. and Güner, E., "A bicriteria flowshop scheduling with a learning effect", ***Applied Mathematical Modelling***, Vol. 32, No. 9, (2008), 1719-1733.
4. Biskup, D., "Single-machine scheduling with learning considerations", ***European Journal of Operational Research***, Vol. 115, No. 1, (1999), 173-178.
5. Biskup, D., "A state-of-the-art review on scheduling with learning effects", ***European Journal of Operational Research***, Vol. 188, No. 2, (2008), 315-329.
6. Chung, Y. and Tong, L., "Make-span minimization for m-machine permutation flow shop scheduling problem with learning considerations", ***Internationl Journal of Advanced Manufacturing Technology***, Vol. 56, No. 1-4, (2011), 355-367.
7. Okołowski, D. and Gawiejnowicz, S., "Exact and heuristic algorithms for parallel-machine scheduling with dejong's learning effect", ***Computers & Industrial Engineering***, Vol. 59, No. 2, (2010), 272-279.
8. Cheng, T., Wu, C.-C., Chen, J.-C., Wu, W.-H. and Cheng, S.-R., "Two-machine flowshop scheduling with a truncated learning function to minimize the makespan", ***International Journal of Production Economics***, Vol. 141, No. 1, (2013), 79-86.
9. Qian, B., Wang, L., Huang, D.-x., Wang, W.-l. and Wang, X., "An effective hybrid de-based algorithm for multi-objective flow shop scheduling with limited buffers", ***Computers & Operations Research***, Vol. 36, No. 1, (2009), 209-233.
10. Sun, Y., Zhang, C., Gao, L. and Wang, X., "Multi-objective optimization algorithms for flow shop scheduling problem: A review and prospects", ***The International Journal of Advanced Manufacturing Technology***, Vol. 55, No. 5-8, (2011), 723-739.
11. Chung, Y. and Tong, L., "Bi-criteria minimization for the permutation flow shop scheduling problem with machine based learning effects", ***Computers & Industrial Engineering***, Vol. 63, No. 1, (2012), 302-312.
12. Corne, D.W., Jerram, N.R., Knowles, J.D. and Oates, M.J., "Pesa-ii: Region-based selection in evolutionary multiobjective optimization", in Proceedings of the Genetic and Evolutionary Computation Conference , GECCO', Citeseer. (2001).
13. Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T., "A fast and elitist multiobjective genetic algorithm: Nsga-ii", ***Evolutionary Computation, IEEE Transactions on***, Vol. 6, No. 2, (2002), 182-197.
14. Suresh, R. and Mohanasundaram, K., "Pareto archived simulated annealing for permutation flow shop scheduling with multiple objectives", in Cybernetics and Intelligent Systems, Conference on, IEEE. Vol. 2, , (2004), 712-717.
15. Varadharajan, T.K. and Rajendran, C., "A multi-objective simulated annealing algorithm for scheduling in flow shops to minimize the make-span and total flow time of jobs", ***European Journal of Operation Research***, Vol. 167, No. 3, (2005), 772-795.
16. Pasupathy, T., Rajendran, C. and Suresh, R., "A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs", ***The International Journal of Advanced Manufacturing Technology***, Vol. 27, No. 7-8, (2006), 804-815.
17. Qian, B., Wang, L., Huang, D.-X. and Wang, X., "Scheduling multi-objective job shops using a memetic algorithm based on differential evolution", ***The International Journal of Advanced Manufacturing Technology***, Vol. 35, No. 9-10, (2008), 1014-1027.
18. Qian, B., Wang, L., Huang, D.-X. and Wang, X., "Multi-objective no-wait flow-shop scheduling with a memetic algorithm based on differential evolution", ***Soft Computing***, Vol. 13, No. 8-9, (2009), 847-869.
19. Ali, M., Siarry, P. and Pant, M., "An efficient differential evolution based algorithm for solving multi-objective optimization problems", ***European Journal of Operational Research***, Vol. 217, No. 2, (2012), 404-416.
20. Storn, R. and Price, K., "Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces, ICSI Berkeley, (1995).
21. Mahmoodabadi, M., Taherkhorsandi, M. and Safikhani, H., "Modeling and hybrid pareto optimization of cyclone separators using group method of data handling (gmdh) and particle swarm optimization (PSO)", ***International Journal of Engineering-Transactions C: Aspects***, Vol. 26, No. 9, (2012), 1089.
22. Namakshenas, M. and Sahraeian, R., "Real-time scheduling of a flexible manufacturing system using a two-phase machine learning algorithm", ***International Journal of Engineering-Transactions C: Aspects***, Vol. 26, No. 9, (2013), 1067.
23. Ghafari, E. and Sahraeian, R., "Appling metaheuristic algorithms on a two stage hybrid flowshop scheduling problem with serial batching", ***International Journal of Engineering***, Vol. 27, No. 6, (2014).
24. Hamta, N., Ghomi, S.F., Bahalke, U. and Golpaigani, H., "Single machine scheduling problem with precedence constraints and deteriorating jobs", ***International Journal of Engineering-Transactions A: Basics***, Vol. 24, No. 2, (2011), 115.

Multi-objective Differential Evolution for the Flow shop Scheduling Problem with a Modified Learning Effect

H. Amirian, R. Sahraeian

Department of Industrial Engineering, College of Engineering, Shahed University, Tehran, Iran

PAPER INFO

Paper history:
Received 26 January 2014
Received in revised form 31 March 2014
Accepted 22 May 2014

Keywords:
Differential Evolution
Multi-Objective Scheduling
Flow shop
Truncated Dejong's Learning Effect

چکیده

در این مقاله الگوریتم چند هدفه‌ی تفاضلی-تکاملی کارآمدی (MDES) برای حل مساله‌ی زمان‌بندی در محیط تولیدی (فلوشاپ) جایگشتی (PFSSP) با در نظر گرفتن اثر یادگیری اصلاح شده‌ی دژونگ ارائه شده است. الگوریتم پیشنهادی، الگوریتم تفاضلی کلاسیک (DE) را با جستجوی محلی ترکیب کرده و در جهت بهبود کارایی کلی، از عملگر انتخاب موجود در NSGA-II استفاده می‌کند. در ابتدا مساله با استفاده از قانون مناسبی کد گذاری شده است تا ماهیت پیوسته الگوریتم تفاضلی را متناسب با مساله محیط تولیدی کند. دوم اینکه، جهت بالا بردن سرعت همگرایی از روش جستجوی محلی بر مبنای جایگذاری در شروع الگوریتم و هم چنین در هر تکرار استفاده شده است. قسمت اول، شروع الگوریتم با جواب‌های بهتر را تضمین می‌کند؛ در حالی که قسمت دوم الگوریتم را روی نواحی نویددهنده متمرکز می‌کند. سوم اینکه، در هر نسل، در جهت بالا بردن تنوع، دو گروه جمعیت معرفی شده است: جمعیت فعلی و جمعیت پیشرفته. بهترین جواب های هر تکرار در جمعیت فعلی ذخیره شده در حالی که جواب‌هایی با مطلوبیت کمتر به جمعیت پیشرفته افزوده می‌شوند. در انتهای هر نسل، دو جمعیت با یکدیگر ترکیب شده و جواب‌های بهتر برای نسل بعدی انتخاب می‌شوند. سپس، در جهت نشان دادن کارایی، الگوریتم بر روی مسائل محک تست شده و نتایج مورد بحث قرار گرفته است. در نهایت نسخه قطع شده‌ای از اثر یادگیری دژونگ پیشنهاد شده و از الگوریتمMDES در جهت حل مساله تولیدی جایگشتی با اثر یادگیری اصلاح شده استفاده شده است.

doi: 10.5829/idosi.ije.2014.27.09c.09