การหาค่าเวลาการทำงานทั้งหมดที่ต่ำที่สุดในปัญหาการจัดตารางการผลิตตามสั่งแบบยืดหยุ่น โดยการปรับแต่งวิธีวิวัฒนาการผลต่าง

วริษา วิสิทธิพานิช*[1]
ภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่
239 ถ.ห้วยแก้ว ต.สุเทพ อ.เมือง จ.เชียงใหม่ 50200

บทคัดย่อ

ปัญหาการจัดตารางงานการผลิตตามสั่งแบบยืดหยุ่น เป็นปัญหาที่ขยายมาจากปัญหาการจัดตารางงานการผลิตตามสั่ง โดยอนุญาตให้การดำเนินการของงานถูกดำเนินการโดยเครื่องจักรหนึ่งตัวที่เลือกจากกลุ่มของเครื่องจักรที่สามารถทำงานนั้นได้ ข้อกำหนดเพิ่มเติมในการเลือกเครื่องจักรที่เหมาะสมในการดำเนินการของแต่ละงานนั้น ทำให้การแก้ปัญหาของการจัดตารางงานการผลิตตามสั่งแบบยืดหยุ่น มีความซับซ้อนมากกว่าการจัดตารางงานการผลิตตามสั่ง งานวิจัยนี้ได้นำเสนอการประยุกต์ใช้วิธีวิวัฒนาการผลต่างที่มีการปรับแต่งเพื่อหาค่าเวลาการทำงานทั้งหมดที่ต่ำที่สุดในปัญหาการจัดตารางการผลิตตามสั่งแบบยืดหยุ่น การปรับแต่งวิธีการแก้ปัญหามีวัตถุประสงค์เพื่อเพิ่มประสิทธิภาพในการหาคำตอบของวิธีวิวัฒนาการผลต่างแบบดั้งเดิม โดยการรักษาสมดุลของความสามารถในการสำรวจหาคำตอบแบบกว้างและการสำรวจหาคำตอบแบบเฉพาะจุด และเพื่อหลีกเลี่ยงปัญหาทั่วไปของการติดอยู่ที่คำตอบใดคำตอบหนึ่งก่อนเวลาอันควร การปรับแต่งวิธีวิวัฒนาการผลต่างวิธีที่หนึ่ง เรียกว่าวิธีวิวัฒนาการผลต่างกับกลยุทธ์การแบ่งกลุ่ม ในวิธีนี้ประชากรจะถูกแบ่งออกเป็นกลุ่มและประชากรแต่ละกลุ่มจะใช้กลยุทธ์ที่ต่างกันในการหาคำตอบใหม่ไปพร้อมๆ กัน เพื่อดึงข้อดีของแต่ละวิธีและชดเชยข้อเสียของแต่ละวิธีและเพิ่มประสิทธิภาพการหาคำตอบโดยรวม การปรับแต่งวิธีวิวัฒนาการผลต่างวิธีที่สอง เรียกว่าวิธีวิวัฒนาการผลต่างกับกลยุทธ์การสลับ ในวิธีนี้ จะให้ประชากรทั้งหมดเปลี่ยนกลยุทธ์ในการหาคำตอบใหม่เมื่อพบว่าคำตอบที่ได้ไม่ดีขึ้น ทำให้โอกาสของการติดอยู่ที่จุดที่คล้ายกับจุดที่ต่ำที่สุดลดลง ประสิทธิภาพของวิธีวิวัฒนาการผลต่างที่มีการปรับแต่งทั้งสองวิธีได้ถูกทดสอบกับปัญหาตัวอย่างและเปรียบเทียบกับคำตอบที่ได้จากวิธีวิวัฒนาการผลต่างแบบดั้งเดิม ผลการทดลองพบว่าวิธีวิวัฒนาการผลต่างที่มีการปรับแต่งทั้งสองวิธีนั้นสามารถหาคำตอบที่ดีเทียบเท่าหรือดีกว่าคำตอบที่ได้จากวิธีวิวัฒนาการผลต่างแบบดั้งเดิม

คำสำคัญ: วิธีการวิวัฒนาการ, วิธีวิวัฒนาการผลต่าง, การผลิตตามสั่งแบบยืดหยุ่น, การจัดตารางงาน, เวลาการทำงานทั้งหมด

* Corresponding author. E-mail: warisa.o@gmail.com

[1] อาจารย์ภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่

Minimizing Makespan in Flexible Job Shop Problems by Adapting the Differential Evolution

Warisa Wisittipanich*[1]
Industrial Engineering Department, Faculty of Engineering, Chiang Mai University,
239 Huay Keaw Rd., Suthep, Muang, Chiang Mai 50200, Thailand

Abstract

The Flexible Job Shop Scheduling Problem (FJSP) is an extension of the classical Job Shop Scheduling Problem (JSP) that allows the operation of a job to be operated on one machine selected from a group of capable machines, and this additional requirement to determine the assignment of operations on proper machines makes the FJSP more complicated than the JSP. This paper discusses the implementation of two adapted differential evolution (DE) algorithms for minimizing makespan in the FJSP. The modified algorithms aim to enhance the efficiency of the original DE by dynamically balancing the exploration and exploitation ability and avoiding the common problem of premature convergence. The first algorithm, called DE with a subgroup strategy, allows the DE population to simultaneously perform different mutation strategies in order to exploit their various strengths and compensate for the weaknesses of each individual strategy in order to enhance overall performance. The second algorithm, called DE with a switching strategy, allows the entire DE population to change the search behavior whenever the solutions do not improve. As a consequence, the chance of getting trapped at a local optimum is reduced. The performances of two adapted DE are evaluated based on a set of benchmark problems and compared with the results obtained from the original DE. The experiment results show that both adapted DE algorithms generate results that are competitive with the original DE.

Keywords: evolutionary algorithm, differential evolution, flexible job shop, scheduling, makespan

* Corresponding author. E-mail: warisa.o@gmail.com
[1] D.Eng in Industrial Engineering Department, Faculty of Engineering, Chiang Mai University

1. Introduction

The classical job shop scheduling problem (JSP) is well-known as one of the most difficult scheduling models and has been the subject of many research efforts for several decades. Although much research work has improved and enhanced the efficiency of solution algorithms for the JSP, the development of efficient solution methods for the large-scale flexible job shop scheduling problem (FJSP) has recently captured the interest of many researchers.

The FJSP is an extension of the classical JSP, which allows an operation of a job to be processed on one machine selected from a set of capable machines. In addition, with the FJSP, some machine stations are allowed to be visited more than once or not at all. Thus, the FJSP is more complex than the JSP due to an additional requirement to determine the assignment of operations on the machines. Generally, the problems of the FJSP can be decomposed into two sub-problems: 1) a routing sub-problem, assigning each operation to a machine selected from a set of capable machines to meet the process requirements; and 2) the scheduling sub-problem, sequencing the assigned operations on each machine in order to obtain a feasible schedule which minimizes the objective function. The FJSP has been proven to be NP-hard, and therefore heuristics approaches are more preferable than traditional exact algorithms for finding high-quality or near-optimal solutions for large-scale problems within a reasonable time.

There is much less literature on the FJSP than on the JSP. However, during the past two decades, metaheuristics have been received increased attention from researchers for solving the FJSP. Typically, the approaches for solving the FJSP are classified into hierarchical and integrated approaches. The hierarchical approach independently considers the assignment of operations to machines and the sequencing of operations on the machines, whereas the assignment and sequencing of operations are considered simultaneously in the integrated approach. The early research on the FJSP was focused on neighborhood-based metaheuristics such as Tabu Search (TS) and Simulated Annealing (SA). Brandimarte [1] introduced a hierarchical approach which combined some dispatching rules and TS in order to minimize makespan in the FJSP. Hurink et al. [2] represented the FJSP as a disjunction graph model and proposed a hierarchical TS-based approach for minimum makespan criteria. Dauzere-Peres and Paulli [3] developed a new disjunction graph model and proposed an integrated approach based on TS to solve the FJSP. Mastrolilli and Gamberdella [4] proposed two new neighborhood functions to improve the performance of Dauzere-Peres and Paulli’s approach. Najid et al. [5] presented a modified SA with a different disjunction graph and neighborhood function in order to minimize makespan in the FJSP. Although neighborhood-based metaheuristics have been successfully applied to solving the FJSP, the performance of these algorithms highly depends on the initial solution and more vulnerable to being trapped into local optima. Consequently, most recent research efforts have been devoted to developing population-based metaheuristics or evolutionary algorithms such as the Genetic Algorithm (GA), Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO). Kacem et al. [6, 7] presented a hierarchical approach for the multi-objective FJSP by applying the GA controlled by the assignment model which was generated by the approach of localization. Zhang and Gen [8] proposed a multistage operation-based GA (moGA) to solve the multi-objective FJSP. Xia and Wu [9] proposed a hybrid PSO and SA to solve the multi-objective

FJSP based on a hierarchical approach where the PSO was used for operation assignment and then the SA was implemented for operation sequencing. Zribi et al. [10] proposed a new hierarchical method using the localization method, and TS and hybrid GA to enhance the quality of solutions. Gao et al. [11] developed a hybrid algorithm for the GA and bottleneck shifting for the multi-objective FJSP in order to fully exploit the global search ability of the GA and the local search ability of the bottleneck shifting heuristic. Li et al. [12] presented an effective hybrid TS (HTSA) with adaptive rules to solve the multi-objective FJSP.

Recently, an evolutionary algorithm called differential evolution (DE) has received attention from many researchers due to its advantage of having relatively few control variables but performing well in search ability and convergence. Although the applications of the DE for combinatorial optimization are still limited, some attempts have been made to apply DE for solving scheduling problems. Godfrey et al. [13] applied DE to the flow shop scheduling problem. Quan et al. [14] proposed a discrete DE algorithm (DDE) for the permutation flow shop with the makespan criteria. Wang et al. [15] proposed a self-adaptive DE (SDE) to improve the global convergence property and to avoid the premature convergence ability of the conventional DE. Liu et al. (2009) extended the application of the DDE to the JSP with special mutation and crossover operators to deal with the discrete variables in the JSP. Warisa and Voratas [16] presented a one-stage DE (1ST-DE), an adaptation of the classical DE, to minimize the makespan for the JSP and the experiments showed that the 1ST-DE was quite competitive with the 1ST-PSO algorithm both in terms of solution quality and solution time. The 1ST-DE was then further applied to minimize the makespan in the FJSP [17]. Later, Warisa and Voratas [18] extended the 1ST-DE to enhance the efficiency of the search by dynamically balancing the exploration and exploitation ability of the DE and avoiding the problem of premature convergence. These two new DE algorithms were applied to minimize two single objective functions: makespan and total weighted tardiness in the JSP. The numerical results demonstrated that the extended DE algorithms yielded promising results while using shorter computing time and fewer numbers of function evaluations compared to those obtained from state-of-the-art PSO algorithms.

This paper discusses the effectiveness of two modified DE algorithms proposed in [18] in order to minimize the makespan in the FJSP. The remainder of this paper is organized as follows. The problem formulation and description of the FJSP are described in section 2. Section 3 describes the classic DE algorithm. Section 4 describes an application of the two modified DE algorithms for the FJSP. The experimental results are reported in section 5. Finally, the conclusion and suggestions for further research are provided in section 6.

2. Problem Description

Similar to the classic JSP, the FJSP schedules a set of n jobs on a set of m machines in order to optimize one or more objectives. However, the problem with the FJSP is that it is more difficult than the JSP because it requires the proper assignment of each job operation to a machine from a set of capable machines. With the FJSP, the set of machines is denoted as M, $M = \{M_1, M_2, \dots, M_m\}$. Each job i consists of a sequence of n_i operations. Each operation $O_{i,j}$ ($i = 1,2,\dots, n : j = 1,2,.., n_i$) of job i needs to be processed on one machine Mk out of a set of given compatible machines $M_{i,j}$. Each machine is independent from others. The machine set up time and the transfer time between operations are negligible. Machine breakdown is not considered and the pre-emption is not allowed in this problem. Each machine can process at most one operation at a time, and there are no precedence constraints among operations of different jobs. The goal is to determine both the assignment and sequence of operations on the machines and to specify the starting time and ending time of each operation in order to optimize certain objectives subjected to constraints.

In this paper, the objective of the model was to minimize the makespan. The notation and variables used in the FJSP model are listed as follows:

n	:	total number of jobs
m	:	total number of machines
n_i	:	total number of operation of job i
$O_{i,j}$	:	the j^{th} operation of job i
$M_{i,j}$	:	the set of available machines for the operation $O_{i,j}$
$P_{i,j,k}$	:	processing time of $O_{i,j}$ on machine k
$S_{i,j,k}$	:	start time of operation $O_{i,j}$ on machine k
$C_{i,j}$	:	completion time of operation $O_{i,j}$
i, h	:	index of jobs ; $i, h = 1, 2, \ldots, n$
k	:	index of machines , $k = 1, 2, \ldots, m$
j,g	:	Index of operation sequence; $j, g = 1, 2, \ldots, n_i$
H	:	A large positive number

$X_{i,j,k}$	=	1 if operation $O_{i,j}$ is assigned to machine k
	=	0 otherwise
$Y_{i,j,i',j',k}$	=	1 if operation $O_{i,j}$ preceded $O_{i',j'}$ on machine k
	=	0 otherwise

The mathematical model of the problem is formulated as follows:

Minimization of makespan:

$$f : minimize\ max\left\{C_{i,j}\right\} \tag{1}$$

Subjected to constraints:

$$C_{i,j} - C_{i,j-1} \geq P_{i,j,k}X_{i,j,k}, j = 2,\ldots,n_i \qquad \forall i,j \tag{2}$$

$$C_{i',j'} - C_{i,j} + H\left(1 - Y_{i,j,i',j',k}\right) + H\left(1 - X_{i,j,k}\right) + H\left(1 - X_{i',j',k}\right) \geq t_{i',j',k} \qquad \forall (i,j),(i',j'),k \tag{3}$$

$$C_{i,j} - C_{i',j'} + H\left(Y_{i,j,i',j',k}\right) + H\left(1 - X_{i,j,k}\right) + H\left(1 - X_{i',j',k}\right) \geq t_{i,j,k} \qquad \forall (i,j),(i',j'),k \tag{4}$$

$$\sum X_{i,j,k} = 1 \qquad \forall i,j \tag{5}$$

$$s_{i,j} \geq 0 \qquad \forall i,j \tag{6}$$

The precedence constraint in equation (2) ensures that an operation of job i must be processed to completion on machine k before the succeeding operation of job i. Equations (3) and (4) present the conflicting constraints to guarantee that each machine can process only one job at a time. Equation (5) states that only one machine can be selected from the set of capable machines for each operation.

3. One-Stage Differential Evolution (1ST-DE)

As mentioned earlier, the 1ST-DE, proposed in [16] is a modification of the original DE. In the 1^{ST} -DE, the initial population was randomly generated, similar to other population-based random searches. In order to obtain a trial vector, a mutation operation was carried out in the same way as with the classic DE. However, the 1ST-DE uses the exponential crossover operation. In the exponential crossover operation, a randomly-picked dimension index *š* is first indicated, and then each dimension value of the trial vector is inherited from its mutant vector, $V_{i,g}$, as long as $u_j \le C_r$. The first time that $u_j > C_r$, all the remaining dimension value are taken from the target vector, $X_{i,g}$. An integer, *L* indicates the number of consecutive dimension indexes on which crossover is performed. The exponential crossover scheme is expressed in equation (7).

$$Z_{j,i,g} = \begin{cases} v_{j,i,g}\ ,if\ j=\langle \check{s}\rangle D,\langle \check{s}+1\rangle D,\ldots,\langle \check{s}+L-1\rangle D \\ x_{j,i,g}\ ,otherwise \end{cases} \quad (7)$$

The angular brackets < >*D* denote a modulo function with modulus D (Storn and Price [19]). The DE variant used in 1ST-DE was denoted as DE/rand/1/exp.

3.1 Adapted DE

This study applied the two modified DE algorithms proposed in [18]. The algorithms aimed to improve the efficiency of the original DE by dynamically balancing exploration and exploitation ability and avoiding the common problem of premature convergence. In the modified DE, a new local mutation operator, named as DE/localbest/1, was introduced and embedded in the algorithms to promote exploitation in different areas of the search space. The procedures of the two modified DE algorithms are explained in the following sections.

3.1.1 DE with subgroup strategy

The DE with a subgroup strategy allows the DE population to simultaneously perform different mutation strategies in order to extract the strengths of various strategies and compensate for the weaknesses of each individual strategy. The purpose of this algorithm is to enhance the overall performance and increase the robustness of the search as a whole. In this algorithm, the DE population was divided into three sub-groups with information sharing across other groups. The DE population in each group executed a particular search strategy with distinct mutation strategies. The population in the first, second, and third group performed the mutation scheme of DE/rand/1, DE/best/1, and DE/localbest/1 respectively. As a result, the combination of multiple search strategies was embedded in one population in order to enhance the overall performance.

3.1.2 DE with switching strategy

The DE with a switching strategy aims to dynamically balance the exploration and exploitation ability of the search and to help the DE avoid becoming trapped at a local optimum through the use of multiple mutation strategies. The concept of this algorithm is to allow the entire DE population changes its search behavior whenever the solutions do not improve. The algorithm employs two mutation strategies: DE/rand/1 and DE/localbest/1. At the beginning, the emphasis is on global exploration (DE/rand/1) in order to let all of the vectors explore the search space vigorously. After some predefined iterations, if the solution is not improved, it implies that it might be trapped at a local optimum, and the search will place new emphasis on the local exploitation (DE/localbest/1) of each vector neighborhood. Again, if the solution is not improved after some predefined iterations, the

|search will be switched back to the first mutation strategy. Consequently, the chance of getting trapped at a local optimum is reduced since the mutation strategies change the movement of the search.

4. Implementation of the DE to the FJSP

In order to apply the DE to a scheduling problem, the solution vectors in the DE need to be transformed into a schedule. A solution of the problem can be represented using a vector with dimensions equal to the total number of operations required for all jobs. The procedures of solution mapping of the DE for the FJSP used in this study are explained using an example of two jobs and three machines, as noted in Table 1.

Table 1 FJSP with 2 jobs and 3 machines

Job	$O_{i,j}$	Processing time		
		M1	M2	M3
1	$O_{1,1}$	3	4	5
	$O_{1,2}$	-	1	2
	$O_{1,3}$	3	6	-
2	$O_{2,1}$	8	7	9
	$O_{2,2}$	-	2	3

According to the example in Table 1, job 1 and job 2 consist of three and two operations respectively. Therefore, the total number of dimensions (d) was set to be equal to the total number of operations of all jobs, which was 5.

Dimension *d*	1	2	3	4	5
Dimension value	0.25	0.89	0.30	0.38	0.67

Figure 1 Random key representation

Figure 1 illustrates the random key representation encoding scheme [20], where each value in a vector dimension was initially generated with a uniform random number in the range [0, 1].

Next, based on the number of operations of each job, this study adopted the permutation of n-repetition of n jobs [21] with a sorting list rule to determine the sequence of operations, as shown in Figure 2. The advantage of this approach is that any permutation of this representation always provides a feasible schedule. Note that this approach has been successfully applied to the JSP in [22] and [23].

Dimension *d*	1	3	4	5	2
Dimension value	0.25	0.30	0.38	0.67	0.89
Job, Operation (*i,j*)	*1,1*	*1,2*	*1,3*	*2,1*	*2,2*

Figure 2 m-repetition of job number permutation and operation-based representation

As mentioned earlier, in the FJSP, an operation is allowed to be processed by one machine selected from a group of capable machines. Therefore, after a sequence of operations is determined, an operating machine for each operation must also be decided. In this study, the machine assigned for each operation was selected based on the earliest completion time of the machine required to process that operation. If two or more machines had the same completion time, one machine was randomly selected as the assigned machine. In order to generate a schedule, the operation-based approach [24] was employed. The decoded individual was transformed into a schedule by taking the first operation from the list of operation sequences, then the second operation, and so on. During the process of generating a schedule, each operation was allocated to an assigned machine having the earliest completion time in the best available position without delaying other scheduled operations. This procedure resulted in completed operation sequences with assigned machines which yielded an active schedule, as shown in figures 3 and 4 respectively.

Dimension *d*	1	2	3	4	5
	0.25	0.89	0.30	0.38	0.67
Job	1	2	1	1	2
machine	1	2	3	1	2

Figure 3 A sequence of operations with assigned machines

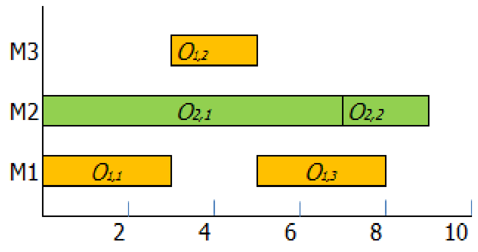

Figure 4 An active schedule after decoding procedure

5. Computational Experiment

5.1 Parameter Setting

In this study, the number of function evaluations was set as 100,000 in order to provide a sufficient search process. Provided by the number of function evaluations, the DE population size and number of iterations were set as 200 and 500 respectively. After some preliminary studies, the value of F was set to be uniformly randomized between 1.5 and 2.5 in order to maintain population diversity throughout the search process. The value of the crossover rate (C_r) was set as linearly increased from 0.1 to 0.5 to preserve the characteristic of the generated trial vectors at the beginning of the search. As the search progressed, the increasing value of C_r provided more variations for the generated trial vectors and helped the solution to escape from being trapped at local optima.

5.2 Numerical Results

The performance of the two adapted DE algorithms was evaluated using three published FJSP data sets: Kacem's data [6],[7], Brandimarte's data [1], and Dauzere-Pere and Paulli's data [3]. Seventeen instances with different problem sizes were selected for the experiments, and each instance was described according to problem size: number of jobs (*n*) x number of machines (*m*) x total number of operations (*O*). Tables 2 shows a statistical comparison of the makespan obtained from two adapted DE algorithms and 1ST-DE. It should be noted that the numerical results reported in the table were obtained from ten independent runs.

It can be seen in Table 2 that most of the solutions obtained from the two adapted DE algorithms were equal to or better than the solutions obtained from the 1ST-DE. For small-size problems, all of the algorithms are capable of finding optimal solutions and all algorithms exhibited their robustness ability. When the problem size increased, all of the algorithms were able to generate good solutions. However, in larger-size instances, the two modified DE algorithms demonstrated superior results compared to the 1ST-DE in terms of both the best makespan value and average makepan value. The better performances of the two adapted DE algorithms were not a consequence of increased computational time. In the two adapted DE algorithms, the distinct features were the changes in the mutation strategy; however, the main evolution process remained the same. As a result, the computing times of the two adapted DE algorithms and 1ST-DE were not significantly different.

Table 2 Comparison of makespan between two adapted DE algorithms and the 1ST-DE

Instance	Size	Best Known Solution	1ST-DE			DE with sub-grouping			DE with strategy switching		
			Best	Avg.	SD	Best	Avg.	SD	Best	Avg.	SD
K01	8 x 8 x 27	14	14	14	0	14	14	0	14	14	0
K02	10 x 10 x 30	7	7	7	0	7	7	0	7	7	0
K03	15 x 10 x 56	11	11	11	0	11	11	0	11	11	0
MK01	10 x 6 x 55	39	40	40	0	40	40	0	40	40	0
MK02	10 x 6 x 58	26	27	27.2	0.42	27	27.1	0.31	27	27.2	0.42
MK05	15 x 4 x 106	172	175	175.1	1.1	173	174.7	1.42	174	174.9	0.87
MK06	10 x 15 x 150	58	62	62.8	1.47	61	62.6	1.07	62	63	0.81
MK07	20 x 50 x 100	140	141	143.4	0.69	142	143.8	0.78	140	143	1.24
MK09	20 x 10 x 203	307	307	307.3	0.94	307	307.5	0.84	307	307.6	1.26
MK10	20 x 15 x 203	198	222	220.6	2.45	217	221.3	2.6	217	221.9	1.91
01a	10 x 5 x 196	2530	2645	2689.2	29.63	2637	2674.7	26.77	2629	2665.7	23.67
04a	10 x 5 x 196	2555	2616	2652.7	14.79	2607	2640.2	21.59	2627	2667.6	22.39
07a	15 x 8 x 293	2396	2582	2609.8	31.72	2521	2584.6	37.40	2566	2601.7	22.75
09a	15 x 8 x 293	2074	2153	2153	10.25	2147	2155.1	6.52	2142	2151.1	9.13
11a	15 x 8 x 293	2078	2221	2240.1	8.71	2212	2233.7	17.89	2216	2236	12.20
16a	20 x 10 x 387	2301	2592	2580.5	27.07	2534	2572.4	23.45	2543	2579.5	25.63
18a	20 x 10 x 387	2139	2236	2219.9	6.98	2216	2225.2	5.67	2215	2224.1	6.83

6. Conclusions

This paper presents the implementation of two adapted DE algorithms in order to minimize makespan regarding the FJSP. The purposes of the algorithms were to improve the performance of the original DE by dynamically balancing exploration and exploitation ability, avoiding premature convergence, and thus increasing the chances to find better solutions. The performances of the two adapted DE algorithms were assessed on a set of benchmark problems and compared with the results from the original DE. The numerical results showed that under the same experimental conditions—the encoding and decoding scheme, and population size and number of iterations— in general, both modified DE algorithms were quite competitive with the original DE in terms of solution quality, especially for the large-size problems.

The ongoing studies have continued to investigate various techniques to improve algorithm performance and robustness for solving a wider range of optimization problems.

7. References

[1] P. Brandimarte, "Routing and scheduling in a flexible job shop by taboo search," *Annals of Operations Research*., Vol. 41, pp. 157–183, 1993.

[2] E. Hurink, B. Jurisch and M. Thole, "Tabu search for the job shop scheduling problem with multi-purpose machines," *Operations Research Spektrum*., Vol. 15, pp. 205–215, 1994.

[3] S. Dauzere-Peres and J. Paulli, "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search," *Annals of Operations Research*., Vol. 70, pp. 281–306, 1997.

[4] M. Mastrolilli and L. M. Gambardella, "Effective neighborhood functions for the flexible job shop problem," *Journal of Scheduling*., Vol. 3 (No.1), pp. 3–20, 2002.

[5] N. M. Najid, S. Dauzere-Peres and A. Zaidat, "A modified simulated annealing method for flexible job shop scheduling problem," *Proceeding of IEEE international conference on systems, man and cybernetics*, 2002.

[6] I. Kacem, S. Hammadi and P. Borne, "Approach by localization and multi-objective evolutionary optimization for flexible job shop scheduling problems," *IEEE Transaction on System, Man and Cybertics, part C*., Vol. 32 (No.1), pp. 1-13, 2002.

[7] I. Kacem, S. Hammadi and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic," *Mathemetics and computers in Simulation*., Vol. 60, pp.245-276, 2002.

[8] H. Zhang and M. Gen, "Multistage-based genetic algorithm for flexible job-shop scheduling problem," *Journal of Complexity International*., Vol. 11, pp. 223-232, 2005.

[9] W. Xia and Z. Wu, "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problem," *Computer and Industrial Engineering*., Vol. 48, pp. 409-425, 2005.

[10] N. Zribi, I. Kacem and A. E. Kamel, "Assignment and Scheduling in flexible job-shop by hierarchical optimization," *IEEE transactions on systems, man and cybernatics, part C: applications and reviews*., Vol. 37 (No.4), pp. 652-661, 2007.

[11] J. Gao, J, M. Gen, L. Sun and X. Zhao, "A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems," *Computer and Industrial Engineering*., Vol. 53, pp. 149-162, 2007.

[12] J. Q. Li, Q. K. Pan and Y.C. Liang, "An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems," *Computer and Industrial Engineering*., Vol. 59, pp. 647-662, 2010.

[13] O. Godfrey and D. Donald, "Scheduling flow shop using differential evolution algorithm," *European Journal of Operational Research*., Vol. 171, pp. 674-692, 2006.

[14] P. Quan-Ke, T. F. Mehmet and L. Yun-Chia, "A discrete differential evolution algorithm for the permutation flowshop scheduling problem," *Proceedings of the 9th Genetic and Evolutionary Computation Conference*, London, pp. 126-133, 2007.

[15] W. Wang, Z. Xiang and X. Xu, "Self-adaptive differential evolution and its application to job-shop scheduling," *Proceeding of the 7th International Conference on System Simulation and Scientific Computing*., pp. 820-826, 2008.

[16] W. Warisa and K. Voratas, "Differential evolution algorithm for job shop scheduling problems," *Industrial Engineering and Management Systems*., Vol. 10 (No.3), pp. 203-208, 2011.

[17] W. Warisa and K. Voratas, "Differential evolution algorithm for makespan minimization in flexible job shop scheduling problem," *Proceeding of the 12th APIEMS*, Beijng, China, 2011.

[18] W. Warisa and K. Voratas, "Two enhanced differential evolution algorithms for job shop scheduling problems," *International Journal of Productions Research*., Vol. 50 (No.10), pp. 2757-2773, 2011.

[19] R. Storn and K. Price, "Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces," *Technical Report TR-95-012, International Computer Science*, Berkeley, CA, 1995.

[20] J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization," *ORSA Journal on Computing*., Vol. 6 (No.2), pp. 154-160, 1994.

[21] C. Bierwirth, In E. Pesch & S. Vo (Eds.), "A generalized permutation approach to job shop scheduling with geneticalgorithms,"*OR-Spektrum. Special issue: Applied Local Search*., Vol. 17 (No. 213), pp. 87-92, 1995.

[22] K. Voratas and S. Siriwan, "A two-Stage genetic algorithm for multi-objective job shop sdcheduling problems," *Journal of Intelligent Manufacturing*., Vol. 22 (No.3), pp. 355-365, 2011.

[23] P. Thongchai and K. Voratas, "A Two-Stage PSO Algorithm for Job Shop Scheduling Problem" *International Journal of Management Science and Engineering Management*., Vol. 6 (No. 2), pp. 84-93, 2011.

[24] R. Cheng, M. Gen and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms – I. representation," *Computers and Industrial Engineering*., Vol. 30, pp. 983-997, 1996.