Differential Evolution Algorithm for Job Shop Scheduling Problem

Article *in* Industrial Engineering & Management Systems · January 2009

DOI: 10.7232/iems.2011.10.3.203

2 authors:

Warisa Wisittipanich
Chiang Mai University
23 PUBLICATIONS **244** CITATIONS
SEE PROFILE

Voratas Kachitvichyanukul
Asian Institute of Technology
95 PUBLICATIONS **2,384** CITATIONS
SEE PROFILE

Differential Evolution Algorithm for Job Shop Scheduling Problem

Warisa Wisittipanich [1]and Voratas Kachitvichyanukul [†2]
Industrial and Manufacturing Engineering
School of Engineering and Technology, Asian Institute of Technology
P.O. Box 4, Klong Luang, Pathumthani 12120, Thailand
Email: warisa.wisittipanich@ait.ac.th[1]
voratas@ait.ac.th[2]

Abstract - *Job shop scheduling is well-known as one of the hardest combinatorial optimization problems and has been demonstrated to be NP-hard problem. In the past decades, several researchers have devoted their effort to develop evolutionary algorithms such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) for job shop scheduling problem. Differential Evolution (DE) algorithm is a more recent evolutionary algorithm which has been widely applied and shown its strength in many application areas. However, the applications of DE on scheduling problems are still limited. This paper proposes a one-stage differential evolution algorithm (1ST-DE) for job shop scheduling problem. The proposed algorithm employs random key representation and permutation of m-job repetition to generate active schedules. The performance of proposed method is evaluated on a set of benchmark problems and compared with results from an existing PSO algorithm. The numerical results demonstrated that the proposed algorithm is able to provide good solutions especially for the large size problems with relatively fast computing time.*

Keywords: *Evolutionary algorithm, Differential Evolution, Scheduling, Job shop*

1. INTRODUCTION

The Job Shop Scheduling Problem (JSP) is one of the most well-known combinatorial problems which have been subject of many research efforts for several decades. The classical JSP schedules a set of jobs on a set of machines in order to optimize a certain criterion, subjected to constraints that each job has processing operations through all machines which are fixed and known in advance. JSP is considered to be NP-hard (Garey *et al*. 1976), and exact algorithms such as integer programming, mixed integer programming and branch and bound method become inefficient to deal with such problem since they can solve only small size problems and may take extremely long computing time when the problem size increases. For practical reasons, most recent efforts have been devoted to develop effective evolutionary algorithms in order to find high-quality or near-optimal solutions within a reasonable time. Several approaches for solving JSP include, but not limited to, Genetic Algorithm GA (Wang *et al*. 2004; Concalves *et al*. 2005; Yamada and Nakano 1995), Ant Colony Optimization ACO (Huang *et al*. 2008; Udomsakdigool and Kachitvichyanukul 2006, 2008) and Particle Swarm Optimization PSO (Pratchayaborirak and Kachivichyanukul 2007; Pongchairerks and Kachivichyanukul 2009).

Differential evolution (DE) is one of the latest evolutionary algorithms developed by Storn and Price in 1995. DE has been gradually receiving attention from researchers due to its advantage of few control variables but performing well in search ability and convergence. It has been widely applied and shown its strengths in many problem domains from scientific, engineering, and financial applications (Price *et al*. 2005; Chakraborty 2008).

However, the applications on DE for combinatorial optimization are still very limited. Recently, some attempts have been made to apply DE for solving scheduling problems. Godfrey *et al.* (2006) applied DE on flow shop scheduling problem. Quan *et al*. (2007) proposed a discrete differential evolution algorithm (DDE) for the permutation flow shop with the makespan criteria. Wang *et al.* (2008) proposed a self-adaptive DE (SDE) to improve global convergence property and avoid premature convergence ability of the conventional DE. Liu *et al*. (2009) extended the application of DDE to the job shop scheduling problem with special mutation and crossover operators to deal with discrete variables in JSP. The results showed that DDE inherits the advantage of the original DE that has a rapid convergence rate and high solution quality for the small size problem, but for large problems DDE lost its diversity causing the solutions to be easily trapped at local optima.

This paper presents the preliminary development of a one-stage differential evolution (1ST-DE) for JSP. The remainders of the paper are organized as follows. Section 2 briefly describes JSP. Section 3 defines the DE algorithm. Section 4 describes an application of the proposed DE for

† : Corresponding Author

JSP. Experimental results are reported in section 5. Finally, conclusion is provided in section 6.

2. JOB SHOP SCHEDULING PROBLEM

The classical JSP is a problem of allocating *n* different jobs to be processed on *m* different machines, subjected to two main sets of constraints; the precedence constraints and the conflict constraints. Each job has a set of sequential operations and each operation must be processed on a predefined machine with deterministic processing time known in advance. Each machine is independent from one another. Machine set up time and transfer time between operations are negligible. In addition, machine breakdown is not considered and the pre-emption is not allowed in this problem. The precedence constraints ensure that each operation of a certain job is processed sequentially and the conflicts constraints guarantee that each machine can processes only one operation at a time. The goal is to sequence operations on machines and specify starting time and ending time of each operation in order to optimize certain objectives subjected to the constraints.

In this paper, the objective is to minimize the makespan. The variables used in the JSP model are listed as follows:

$p_{j,k}$	: the process time of job j on machine k
$s_{j,k}$	: the start time of job j on machine k
M	: a large positive number
r_j	: the ready time of job j
$y_{j,j',k}$	: a binary variable defined as

$$y_{j,j',k} = \begin{cases} 1, \text{if job } j \text{ is before job } j' \text{on machine } k \\ 0, \text{ otherwise} \end{cases}$$

The mathematical model of the problem can be formulated as follows:

Minimization of makespan

$$f: minimize\ max\ \{s_{j,k} + p_{j,k}\} \qquad (1)$$

Subjected to

$$s_{j,k} + p_{j,k} \leq s_{k'} \qquad \forall j,k,k' \qquad (2)$$

$$s_{j,k} + p_{j,k} \leq s_{j',k} + M.(1 - y_{j,j',k}) \qquad \forall j,j',k \qquad (3)$$

$$s_{j',k} + p_{j',k} \leq s_{j,k} + M.y_{j,j',k} \qquad \forall j,j',k \qquad (4)$$

$$s_{j,k} \geq r_j \geq 0 \qquad \forall j,k \qquad (5)$$

Where j, j'= {1, 2, ..., *n*} and k, k'={1,2,…, *m*}

Equation (1) is the objective function. Equation (2) is the precedence constraint, Equations (3) and (4) are the conflict constraints. Equation (5) ensures that any job cannot start before its ready time.

3. DIFFERENTIAL EVOLUTION

Differential Evolution (DE) is one of the Evolutionary Algorithms (EAs) for global optimization over continuous search space (Storn and Price 1995). Its theoretical framework is simple and requires inexpensive computation in term of CPU time (Bin *et al*. 2008). Due to its advantage of relatively few control variables but performing well in convergence, DE has been widely applied and shown its strengths in many application areas (Godfrey and Donald 2006; Quan *et al*. 2007; Qian *et al*. 2008).

As a population-based search method, DE starts with randomly generate initial population of size *N* of *D*-dimensional vectors. A solution in DE algorithm is represented by *D*-dimensional position of a vector. Each variable's value in the dimensional space is represented as the real number. The key idea behind DE is a new mechanism for generating trial vectors. DE generates trial vectors by mutation and crossover operation. Then, the replacement of an individual, so called selection operation, occurs only if the trial vector outperforms its corresponding vector. These processes are repeated until some stopping criteria are met. Thus, DE population evolve through repeated cycles of three main DE operators; mutation, crossover and selection.

3.1 The Classic DE

Currently, there are several DE variants. Generally, the mutation is the main operation which makes each variant distinct from one another. The classic version of DE is the simplest and most popular scheme used in literatures. The mutation operator in the classic DE generates a mutant vector by adding a weighted difference between two randomly selected vectors to the third randomly selected vector. This scheme has proven to be effective in solving many multimodal optimization problems due to its good exploration capability (Price *et al*. 2005; Chakraborty 2008). The procedures in the classic DE (Price *et al*. 2005) can be described as follows:

3.1.1 Population Initialization

DE starts with initializing the population of size *N* of *D*-dimensional vectors. The lower bound, b_L, and upper bound, b_U, for the value in each dimension j^{th} (j=0,1,…,D-1) must be specified. At initialization step (g =0), the j^{th} value of the i^{th} vector is randomly generated as follows:

$$x_{j,i,0} = u_j.(b_{j,U} - b_{j,L}) + b_{j,L} \quad (6)$$

Where u_j is a uniform random number in the range [0, 1].

3.1.2 Mutation

Once initialized, DE mutates and combines current target vectors to produce mutant vectors. For each target vector, $X_{i,g}$, at generation g, the mutant vector, $V_{i,g}$, is generated according to the following equation:

$$V_{i,g} = X_{r1,g} + F(X_{r2,g} - X_{r3,g}) \quad (7)$$

It is noted that X_{r1}, X_{r2}, and X_{r3} are randomly selected vectors from the population. They are mutually exclusive and different from the i^{th} target vector, $X_{i,g}$. F is a scale factor which controls the scale of the difference vector between X_{r2}, and X_{r3}, added to the base vector, X_{r1}.

3.1.3 Crossover

DE applies crossover operator on $X_{i,g}$ and $V_{i,g}$ to generate the trial vector $Z_{i,g}$. In the classic DE, the uniform crossover is employed and the trial vector is generated by the following equation.

$$z_{j,i,g} = \begin{cases} v_{j,i,g}, & \text{if } u_j \le C_r \text{ or } j = j_u \\ x_{j,i,g}, & \text{otherwise} \end{cases} \quad (8)$$

Where

u_j : a uniformly random number between [0, 1]

j_u : a random chosen index, $j_u \in \{0, 1, \ldots D-1\}$

C_r : crossover probability in the range [0, 1]

C_r controls the probability of selecting the value in each dimension for a trial vector from a mutant vector.

3.1.4 Selection

The selection operation is performed on each target vector, $X_{i,g}$,and its corresponding trial vector, $Z_{i,g}$, to determine the survival vector for the next generation. The vector, $X_{i,g+1}$, is selected according to the greedy criteria

$$X_{i,g+1} = \begin{cases} Z_{i,g}, & \text{if } f(Z_{i,g}) \le f(X_{i,g}) \\ X_{i,g}, & \text{otherwise} \end{cases} \quad (9)$$

It is noted that this selection scheme is applicable for minimization problem. Once each individual in the current population is updated, the population continue to evolve through mutation, crossover, and selection operation until some stopping criteria are met.

This classic DE is commonly denoted as DE/rand/1/bin, where DE stands for DE, rand is the type of base vector selected to be perturbed, 1 is the number of difference vector for permutation, and bin stands for binomial distribution of the number of inherited dimension values of mutant vectors.

4. APPLICATION OF DE ON JSP

Since DE is first designed for continuous domain, in order to apply DE to combinatorial problem i.e. scheduling problem, solution vectors in DE need to be transformed into a schedule. The procedures of solution mapping are illustrated in the following example.

4.1 Solution Representation

A solution of the problem can be represented using a vector with dimensions equal to the total number of operations processed on all machines. Consider an example of two jobs and three machines in JSP in table 1.

Table 1: JSP with 2 jobs and 3 machines

Job	Machine Sequence			Processing Time		
A	M1	M2	M3	3	2	6
B	M2	M1	M3	5	3	4

According to this example, the total number of dimension, equal to the total number of operation, is 6. Figure 1 illustrates random key representation encoding scheme (Bean 1994) where each value in a vector dimension is initially generated with a uniform random number in range [0, 1].

Dimension j	1	2	3	4	5	6
	0.23	0.97	0.34	0.46	0.71	0.58

Figure 1: Random key representation

Next, the permutation of 3-repetition of 3 jobs (Bierwirth 1995) is applied with a sorting list rule to decode an individual vector into a 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. Then, the operation-based approach (Cheng *et al.* 1996) is employed to generate a schedule. The decoded individual is transformed into a schedule by taking the first operation from the list, then the second operation, and so on. During the process of generating a schedule, each operation is allocated to a required machine in the best available position without delaying other scheduled operations. This procedure results in an active schedule as shown in figure 3.

Dimension j	1	3	4	6	5	2
	0.23	0.34	0.46	0.58	0.71	0.97
	A	A	A	B	B	B

Dimension j	1	2	3	4	5	6
	0.23	0.97	0.34	0.46	0.71	0.58
	A	B	A	A	B	B

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

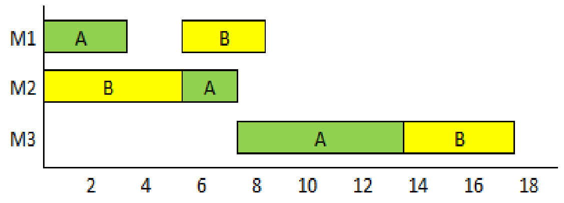

Figure 3: An active schedule after the decoding process

4.2 A Single Stage DE (1ST- DE)

This paper proposes a one-stage differential evolution (1ST-DE) with a single population. In 1ST-DE, initial population is randomly generated according to equation (6).To obtain trial vectors, the mutation operation is carried out in the same way as the classic DE by equation (7), however, this study uses the exponential crossover operation. Starting at the randomly picked dimension index $\check{s}$, each dimension value of trial vector are inherited from its mutant vector, $V_{i,g}$, as long as $u_j \leq C_r$. The first time when $u_j > C_r$, 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 can be expressed in equation (10).

$$z_{j,i,g} = \begin{cases} v_{j,i,g}\text{, if } j=\langle \check{s} \rangle_D, \langle \check{s}+1 \rangle_D, \ldots, \langle \check{s}+L-1 \rangle_D \\ x_{j,i,g}\text{, otherwise} \end{cases} \qquad (10)$$

The angular brakets $<\ >_D$ denote a modulo function with modulus D (Storn and Price 1995). The variant used in the experiments of 1ST-DE can be denoted as DE/rand/1/exp.

5. COMPUTATIONAL EXPERIMENT

5.1 Parameter Setting

In this study, the crossover rate C_r is set to 0.2. Some experiments were carried out to determine the choice of scale factors F in the range [0, 2]; 0.5, 1, 1.5, and 2. The results showed that the suitable value of F for JSP in this study is 1.5. Particularly compared with PSO, DE population size and number of iterations are set equally to those in previous PSO studies as 100 and 500 respectively (Pratchayaborirak and Kachitvichyanukul 2007).

5.2 Experimental Results

The experiments are implemented using the C# language of the Microsoft Visual Studio 8.0. The program runs on the platform of Intel®Core^{TM} 2 Dou CPU 1.67GHz with 3062 MRAM.

The performance of 1ST-DE is evaluated using several benchmark JSP; FA20, LA01, LA21, LA29, LA35 and LA37 representing different problem sizes. The results are compared to those obtained from 1ST-PSO (a one-stage PSO) having the same algorithm as in previous PSO studies (Pratchayaborirak and Kachitvichyanukul 2007) under the same conditions such as encoding and decoding scheme, population size and number of iterations. Table 2 shows the best, average, standard deviations of makespan, and computing time from 10 runs of each algorithm for each problem. The t-test was performed to statistically compare the results of two algorithms with 95% confident interval. The average convergence graph are also observed and illustrated in figure 4.

Table 2: Makespan comparison results between 1ST-PSO and 1ST-DE

Instance	Problem size	Best known Solution	1ST-PSO				1ST-DE				P-Value
			Fitness Value			Time	Fitness Value			Time	
			Best	Avg.	SD	(Sec.)	Best	Avg.	SD	(Sec.)	
FT20	10x10	1165	1226	1279.9	29.516	12.82	1224	1255	19.686	6.878	0.042
LA01	5x10	666	666	666	0	4.85	666	666	0	2.437	1
LA21	15x10	1046	1159	1167.4	6.501	20.36	1126	1141.1	9.539	8.352	0.000
LA29	20x10	1152	1319	1335.4	9.489	29.91	1277	1300.7	13.329	13.857	0.000
LA35	30x10	1888	1939	1967.6	20.727	58.03	1899	1917	12.875	26.52	0.000
LA37	15x15	1397	1514	1545.4	16.44	28.52	1485	1515.8	20.368	14.76	0.002

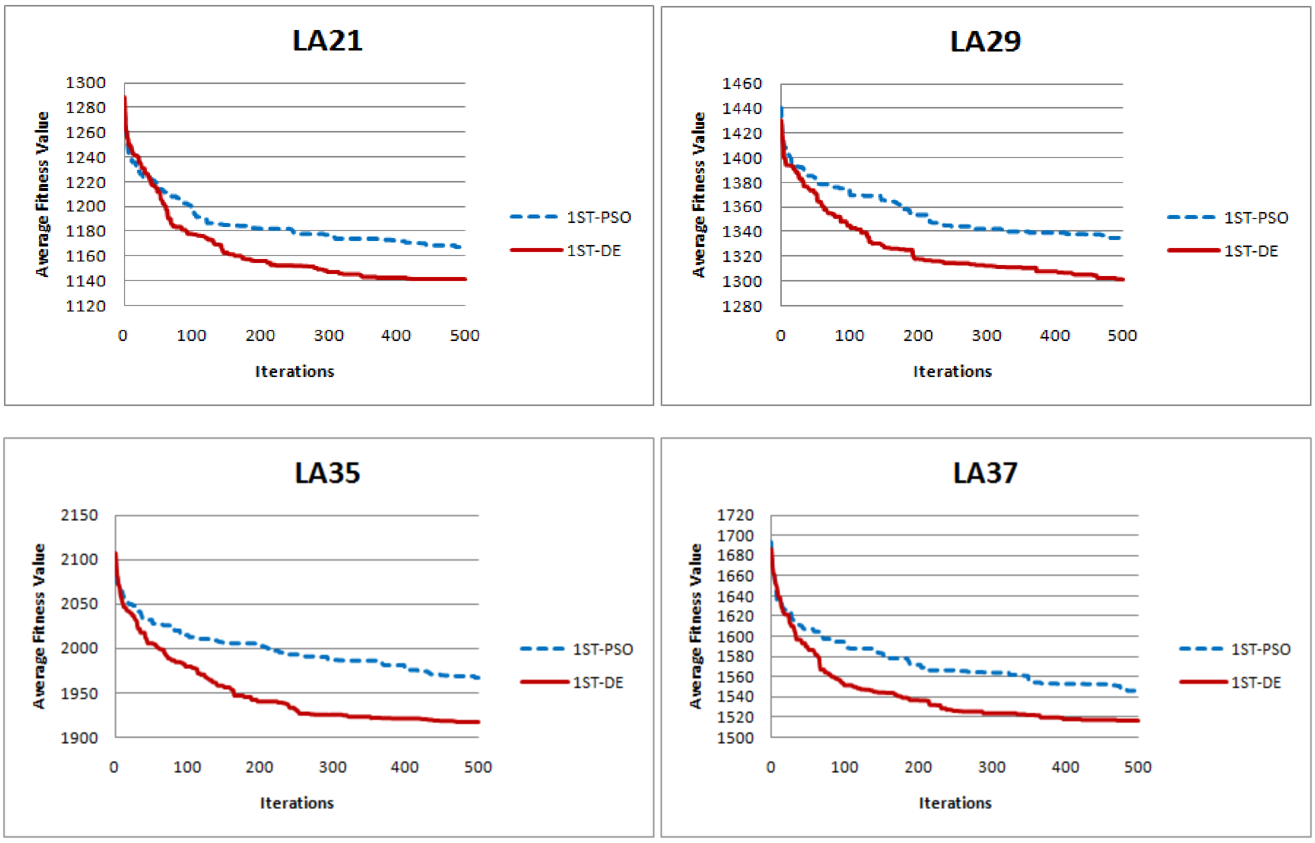

Figure 4: Comparison of average convergence graph between PSO and 1ST-DE

It is noted that since this experiment focuses on the performance of DE and PSO algorithms, the 1ST-DE and 1ST-PSO used here are the simple versions without any additional improvements such as re-initialization and local search strategies.

According to the results from table 2 and figure 4, it is observed that all of solutions obtained from the proposed DE are equal to or better than those obtained from 1ST-PSO with relatively faster computing time. The statistical t-test indicates that 1ST-DE outperforms 1ST-PSO in all cases except for small problem size LA01 where both algorithms can easily find the optimal solution. The average convergence graph clearly demonstrates faster convergence behavior of 1ST-DE compared to 1ST-PSO.

6. CONCLUSION

This paper proposes a one-stage differential algorithm (1ST-DE) for solving JSP. The proposed algorithm employs the classic DE mutation scheme with exponential crossover operation. The random key representation and permutation of m-job repetition is applied to generate an active schedule. The performance of proposed method is evaluated on a set of benchmark problems and compared with results from an existing PSO algorithm.

The experimental results indicate that under the same experimental conditions: encoding and decoding scheme, population size and number of iterations, the 1ST-DE is superior to 1ST-PSO in term of solution quality and computing time especially for the large size problems. In addition, the proposed DE demonstrates faster convergence behavior than that in 1ST-PSO.

In conclusion, it can be claimed that DE has mechanisms to effectively search for better solutions than PSO. The ongoing researches are under investigation to improve DE performance for solving a wide range on scheduling problems.

REFERENCES

Bean, J. C. (1994) Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, **6**(2), 154-160.

Bierwirth, C. (1995) In E. Pesch, & S. Vo (Eds.), A generalized permutation approach to job shop scheduling with genetic algorithms. *OR-Spektrum. Special issue: Applied Local Search*, **17**(213), 87-92.

Bin Q., Ling W., De-Xian H., and Xiong W. (2008) Scheduling multi-objective job shop using a memetic algorithm based on differential evolution. *International Journal of Advanced Manufacturing and Technology*, **35**, 1014-1027.

Chakraborty, U.K. (ed.) (2008) *Advances in Differential Evolution*, Springer, Heidelberg.

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

Fang L., Yutao Q., Zhuchang X., and Hongxia H. (2009) Discrete differential evolution for the job shop scheduling problem. *Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionaty Computation, China*, 879-882.

Garey, M. R., Johnson, D. S., and Sethi, R. (1976) The complexity of flow shop and job-shop scheduling. *Mathematics of Operation Research*, **1**, 117-129.

Godfrey, O. and Donald, D. (2006) Scheduling flow shop using differential evolution algorithm. *European Journal of Operational Research*. **171**, 674-692.

Gonçalves, J. F., José, J., and Resende, M.G.C. (2005) A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operation Research*, **167**, 77-95.

Huang, K. L. and Liao, C. J. (2008) Ant colony optimization combined with taboo search for the job shop scheduling problem. *Computer and Operations Research*, **35**, 1030-1046.

Price, K., Storn, R.M., and Lampinen, J.A. (2005) *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Springer, New York.

Pongchairerks, P. and Kachitvichyanukul, V. (2009) A two-level particle swarm optimization algorithm on job-shop scheduling problems. *International Journal of Operational Research*, **4**(4), 390-411.

Pratchayaborirak, T., and Kachivichyanukul, V., (2007). A two-stage particle swarm optimization for multi-objective job shop scheduling problems. *Proceedings of the 8th Asia Pacific Industrial Engineering and Management Systems Conference (APIEMS), Taiwan*.

Qian, B., Wang, L., Huang, D. X., and Wang, X. (2008) Scheduling multi-objective job shops using memetic algorithm based on differential evolution. *International Journal of Advanced Manufacturing and Technology*, **35**, 1014-1027.

Quan-Ke, P., M. Fatih, T., and Yun-Chia, L. (2007) A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Proceedings of the 9th Genetic and Evolutionary Computation Conference, London*, 126-133.

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

Udomsakdigool, A. and Kachitvichyanukul, V. (2006) Two-way scheduling approach in ant algorithm for solving job shop problems. *International Journal of Industrial Engineering and Management Systems*, **5**(2), 68-75.

Udomsakdigool, A. and Kachitvichyanukul, V. (2008) Multiple-colony ant algorithm with forward–backward scheduling approach for job-shop scheduling problem. *Advances in Industrial Engineering and Operation Research (Springle)*, chapter 4, 39-55.

Wang, W. L., Wu, Q. D. and Song, Y. (2004) Modified adaptive genetic algorithms for solving job-shop scheduling problems. *System Engineering Theory and Practice*, **24**(2), 58-62.

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

Yamada, T. and Nakano, R. (1995) A genetic algorithm with multi-step crossover for job-shop scheduling problems. *Proceedings of the IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, 146-151.

AUTHOR BIOGRAPHIES

Warisa Wisittipanich is a Doctoral candidate in Industrial and Manufacturing Engineering, Asian Institute of Technology, Thailand. She received Master Degree in Systems Engineering from George Mason University, Virginia, USA. Her research interests include scheduling and evolutionary method for optimization. She can be reached at warisa.wisittipanich@ait.ac.th

Voratas Kachitvichyanukul is a Professor of Industrial and Manufacturing Engineering, Asian Institute of Technology, Thailand. His teaching and research interests include modeling and simulation of large scale industrial systems, evolutionary methods for optimization, supply chain management and applied operations research. He can be reached at voratas@ait.ac.th