International Journal of Operations Research

International Journal of Operations Research Vol. 3, No. 2, 120-135 (2006)

A Particle Swarm Optimization and Differential Evolution Algorithms for Job Shop Scheduling Problem

M. Fatih Tasgetiren[1], Mehmet Sevkli[2,*], Yun-Chia Liang[3], and M. Mutlu Yenisey[4]

[1]Department of Operations Management and Business Statistics, Sultan Qaboos University, P.O. Box 20, Al Khod 123, Muscat Sultanate of Oman

[2]Department of Industrial Engineering, Fatih University, 34500 Buyukcekmece, Istanbul, Turkey

[3]Department of Industrial Engineering and Management, Yuan Ze University, No. 135 Yuan-Tung Road, Chung-Li, Taoyuan County, Taiwan 320, R.O.C.

[4]Department of Industrial Engineering, Istanbul Technical University, 34367, Macka, Istanbul, Turkey

Abstract—In this paper, we present particle swarm optimization (PSO) and differential evolution (DE) algorithms for the job shop scheduling problem with the makespan criterion. The applications of PSO and DE on combinatorial optimization problems are still considered limited, but the advantages of PSO and DE algorithms such as structural simplicity, accessibility to practical applications, ease of implementation, speed to get the solutions, and robustness are already shown in the literature. However, the major obstacle of successfully applying PSO and DE algorithms to combinatorial optimization problems is due to their continuous nature. To remedy this drawback, the smallest position value (SPV) rule presented in Tasgetiren et al.(2004a, b, c, d) is employed in both algorithms to convert continuous position values to discrete job permutations. In order to improve the solution quality, both algorithms are also hybridized with an efficient local search method based on a variable neighborhood search (VNS) technique. The experimental results based on the well known benchmark instances collected from OR library show that the hybrid PSO algorithm has generated slightly better results than its counterpart, namely, the DE algorithm. It is also shown that the hybrid PSO algorithm is either better or competitive to the state-of-the-art methods in the literature. In addition, to the best of our knowledge, both algorithms are the first reported applications of PSO and DE algorithms for the job shop scheduling problem in the literature.

Keywords—Particle swarm optimization, Differential evolution, Job shop scheduling, Makespan, Variable neighborhood search

1. INTRODUCTION

Scheduling, a form of decision-making, plays an essential role in manufacturing as well as in the service industry. A corporation must respond quickly and precisely to the customers' demands in order to maintain market share. Thus, effective and efficient scheduling has become a necessity for survival in the modern competitive marketplace. Among the typical goals of scheduling problems, maximizing machine utilization is not only a measure of academic interest, but also useful and important in practice. When considering the job shop scheduling (JSS) problem, the general form of the classical scheduling problems, the goal above can be intuitively transferred to makespan minimization. Therefore, the JSS problem with the objective function of minimizing makespan can be stated as follows. Each of n jobs is to be processed without preemption by m machines. Each job consists of m operations that own a predetermined processing order through machines. Each machine can handle no more than one job at a time and each job must visit each machine only once. The release time of all jobs is zero. Set-up and knock-down times on each machine are included in the processing time. Then, the JSS problem is to schedule jobs that minimizes the maximum completion time over all jobs, i.e.,

$$C_{\max} = \max_{j=1,..,n} \{C_j\} \tag{1}$$

where C_j is the completion time of job j, for $j = 1, 2, ..., n$. For the computational complexity of the JSS problem, Garey et al. (1976) proved that it is NP-hard. Small size instances of the JSS problem can be solved with reasonable computational time by exact algorithms such as branch-and-bound (Carlier and Pison, 1989; Applegate and Cook, 1991), and the time orientation approach (Martin, 1996). However, when the problem size increases, the computational time of exact methods grows exponentially. On the other hand heuristic algorithms have generally acceptable time and memory requirements, but do not guarantee optimality of the final solution, that is, a feasible

* Corresponding author's email: msevkli@fatih.edu.tr

solution is obtained which is likely to be either optimal or near-optimal. Therefore, the most recent research on JSS problems has been focused on heuristic algorithms. The solution techniques of heuristic algorithms can be broadly classified into two groups: meta-heuristics and local search type heuristics. In the first category, Simulated Annealing (SA) (Van Laarhoven et al., 1992; Yamada and Nakano, 1996; Satake et al., 1999; Steinhöfel et al., 1999, 2002; Aydin and Fogarty, 2004), Genetic Algorithm (GA) (Yamada and Nakano, 1992; Bierwith, 1995; Groce et al., 1995; Dorndorf and Pesch, 1995; Ikeda and Kobayashi, 2000; Murovec and Šuhel, 2004), Tabu Search (TS) (Dell'Amico and Trubian, 1993; Taillard, 1994; Nowicki and Smutnicki, 1996; Pezzella and Merelli, 2000; Murovec and Šuhel, 2004), Ant Colony Optimization (ACO) (Colorni et al., 1994; Blum and Sampels, 2004), hybrid SA and GA (Kolonko, 1999; Wang and Zheng, 2001), Neural Network (NN) (Zhou et al., 1991; Satake et al., 1994) have provided abundant research. The latter group consists of shifting bottleneck procedure (Adams et al., 1988; Huang and Yin, 2004), guided local search (Balas and Vazacopoulos, 1998), constraint propagation (Brinkkötter and Brucker, 2001; Dorndorf et al. 2002), and parallel Greedy Randomized Adaptive Search Procedure (GRASP) (Aiex et al., 2003). The comprehensive survey of the JSS problem can be found in Aarts and Lenstra (1997), Blazewicz et al. (1996), and Jain and Meeran (1999).

Particle Swarm Optimization (PSO) and Differential Evolution (DE) are two of the latest metaheuristic methods. PSO is based on the metaphor of social interaction and communication such as bird flocking and fish schooling. PSO is different from other evolutionary-type methods in a way that it does not use the filtering operation (such as crossover and/or mutation), and the members of the entire population are maintained through the search procedure so that information is socially shared among individuals to direct the search towards the best position in the search space. In a PSO algorithm, each member is called a *particle*, and each particle moves around in the multi-dimensional search space with a velocity constantly updated by the particle's experience, the experience of the particle's neighbors, and the experience of the whole swarm. Like the real-coded PSO algorithm, candidate solutions in DE are represented as individuals based on floating-point numbers. In the DE algorithm, the target population is perturbed with a mutant factor, and the crossover operator is then introduced to combine the mutated population with the target population so as to generate a trial population. Then the selection operator is applied to compare the fitness function value of both competing populations, namely, target and trial populations. The better individuals among these two populations become members of the population for the next generation. This process is repeated until a convergence occurs.

PSO and DE were both first introduced to optimize various continuous nonlinear functions by Eberhart and Kennedy (1995) and Storn and Price (1995, 1997), respectively. PSO has been successfully applied to a wide range of applications such as automated drilling (Onwubolu and Clerc 2004), lot sizing problems (Tasgetiren and Liang 2003), mass-spring systems (Brandstatter and Baumgartner 2002), neural network training (Van den Bergh and Engelbecht 2000), permutation flowshop sequencing problems (Tasgetiren et al. 2004a, 2004d), power and voltage control (Yoshida et al., 2000; Abido, 2002), single machine total weighted tardiness problems (Tasgetiren et al., 2004c), supplier selection and ordering problems (Yeh, 2003), and task assignment (Salman et al., 2003). DE's applications consist of aerodynamic design (Rogalsky et al., 2000), digital filter design (Storn, 1999), earthquake relocation (Ruzek and Kvasnicka, 2001), microprocessor synthesis (Rae and Parameswaran, 2001), neural network learning (Masters and Land, 1997), and permutation flowshop sequencing problems (Tasgetiren et al., 2004b). More comprehensive surveys of PSO can be found in Kennedy et al. (2001), DE in Lampinen (2001), and Onwubolu and Babu (2004).

The applications of PSO and DE on combinatorial optimization problems are still considered limited, but the advantages of PSO and DE algorithms such as structural simplicity, accessibility to practical applications, ease of implementation, speed to get the solutions, and robustness are shown in the literature. However, the major obstacle of successfully applying PSO and DE algorithms to combinatorial problems in the literature is due to their continuous nature. To remedy this drawback, Tasgetiren et al. (2004a, 2004b, 2004c and 2004d) present the smallest position value (SPV) rule, borrowed from the *random key representation* of (Bean, 1994), for the PSO and DE algorithms to convert a continuous position vector to a discrete job permutation. This has been effectively applied to the single machine total weighted tardiness (SMTWT) problem and the permutation flowshop sequencing problem (PFSP). Following the successful applications above, this paper aims at employing PSO and DE in solving the job shop scheduling problem.

The organization of this paper is as follows. Section 2 and 3 introduce the PSO and DE algorithms respectively. Neighborhood structure of both algorithms is presented in Section 4 followed by experimental results given in Section 5. Finally, Section 6 summarizes the concluding remarks.

2. PARTICLE SWARM OPTIMIZATION ALGORITHM FOR JSS PROBLEM

In a PSO algorithm for the JSS problem, the initial population is generated randomly. Then the SPV rule for each particle is used to convert the continuous position values to its permutation of operations. Then the job repetition vector is obtained to evaluate the fitness value, the makespan, of the particle. After evaluation, the PSO algorithm repeats the following steps iteratively.

Initially, each individual with its position, velocity, and fitness value is assigned to its personal best (i.e., the best value of each individual found so far). The best individual in the whole swarm with its position and fitness value, is, on the other hand, assigned to the global best (i.e., the best particle in the whole swarm). Then each particle updates its

velocity based on the experiences of the personal best and the global best in order to update the position of each particle with the velocity currently updated. Corresponding permutation of operations and job repetition are determined through the SPV rule so that an evaluation is again performed to compute the fitness of the particles in the swarm. In addition, a local search may apply to a certain group of particles in the swarm to enhance the exploitation of search space. This process is terminated with a predetermined stopping criterion. This study follows the *gbest* model of Eberhart and Kennedy (1995) with the inclusion of the SPV rule in the algorithm. Pseudo code of the PSO algorithm for the JSS problem is given in Figure 1.

Initialize parameters
Initialize population
Find permutation of operations
Find job repetition
Evaluate
Do {
  Find personal best
  Find global best
  Update velocity
  Update position
  Find permutation of operations
  Find job repetition
  Evaluate
  Apply local search
  } *While (Not Termination)*

Figure 1. PSO algorithm with local search for JSS problem.

The basic elements of PSO algorithm is summarized as follows:

Particle: X_i^t denotes the i^{th} particle in the swarm at iteration t and is defined as $X_i^t = [x_{i1}^t, x_{i2}^t, ..., x_{i,nm}^t]$, where x_{ik}^t is the position value of the i^{th} particle with respect to the k^{th} dimension (k = 1, 2, ..., nm).

Population: X^t is the set of NP particles in the swarm at iteration t, i.e., $X^t = [X_1^t, X_2^t, ..., X_{NP}^t]$ where NP denotes the population size.

Permutation of operations: A new variable φ_i^t, which is a permutation of operations of jobs implied by the particle X_i^t is introduced. It can be described as $\varphi_i^t = [\varphi_{i1}^t, \varphi_{i2}^t, ..., \varphi_{i,nm}^t]$, where φ_{ik}^t is the assignment of operation k of the particle i in the permutation of operations at iteration t.

Job repetition: Another variable π_i^t, which is a repetition of jobs implied by the particle X_i^t can be described as $\pi_i^t = [\pi_{i1}^t, \pi_{i2}^t, ..., \pi_{i,nm}^t]$, where π_{ik}^t is the assignment of job j of the particle i repeating m times in the job repetition vector at iteration t.

Particle velocity: V_i^t is the velocity of particle i at iteration t and is defined as $V_i^t = [v_{i1}^t, v_{i2}^t, ..., v_{i,nm}^t]$, where v_{ik}^t is the velocity of particle i at iteration t with respect to the k^{th} dimension (k = 1, 2, ..., nm).

Inertia weight: w^t is a parameter to control the impact of the previous velocities on the current velocity.

Fitness function: In a minimization problem, the objective function is $f_i\left(\pi_i^t \leftarrow X_i^t\right)$ where π_i^t is the corresponding job repetition vector of particle X_i^t.

Personal best: P_i^t represents the best position of the particle i with the best fitness value until iteration t, and is called the *personal best*. For each particle in the swarm, P_i^t can be determined and updated at each iteration t. In a minimization problem with the objective function $f(\pi_i^t \leftarrow X_i^t)$, the personal best P_i^t of the i^{th} particle is obtained such that $f(\pi_i^t \leftarrow P_i^t) \le f(\pi_i^{t-1} \leftarrow P_i^{t-1})$ for i = 1, 2, ..., NP. To simplify, we denote the fitness function of the personal best as $f_i^{pb} = f(\pi_i^t \leftarrow P_i^t)$. For each particle, the personal best is defined as $P_i^t = [p_{i1}^t, p_{i2}^t, ..., p_{i,nm}^t]$ where p_{ik}^t denotes the position value of the i^{th} personal best with respect to the k^{th} dimension (k = 1, 2, ..., nm).

Global best: G^t denotes the best position of the globally best particle achieved so far in the whole swarm. Therefore, the global best can be obtained such that $f(\pi^t \leftarrow G^t) \le f(\pi_i^t \leftarrow P_i^t)$ for i = 1, 2, ..., NP. To simplify, the fitness function of the global best is denoted as $f^{gb} = f(\pi^t \leftarrow G^t)$. The global best is then defined as $G^t = [g_1^t, g_2^t, ..., g_{nm}^t]$ where g_k^t is the position value of the global best with respect to the k^{th} dimension (k = 1, 2, ..., nm).

2.1 Solution representation

In this paper, an operation-based representation of Cheng et al. (1996) was used encoding a schedule as a repetition of jobs. Each dimension represents one operation of a job appearing exactly m times in the job repetition vector π_i^t. For the n-job and m-machine problem, each particle contains $n \times m$ number of dimensions corresponding to $n \times m$ operations.

Particles have a continuous set of values for its dimensions. The particle itself does not present a solution. Instead, the smallest position value, the SPV rule of Tasgetiren et al. (2004a, 2004b, 2004c, and 2004d) is used first to find the permutation of operations. Then the job repetition vector is determined by the following formula:

$$\pi_{ik}^t = \left\lfloor \frac{\varphi_{ik}^t - 1}{n} \right\rfloor + 1. \qquad (2)$$

Table 1 illustrates the solution representation of particle X_i^t of the PSO algorithm for the 3-job 3-machine problem containing 3 × 3 = 9 operations. It should be noted that the SPV rule converts the continuous position values to a discrete job permutation which is the key to

enable the continuous PSO algorithms to be applied to sequencing problems. According to the SPV rule, the smallest position value is $x_{i6}^t = -2.25$, so the dimension k = 6 is assigned to be the first job $\varphi_{i1}^t = 6$ in the permutation φ_i^t; the second smallest position value is $x_{i2}^t = -0.99$, so the dimension k = 2 is assigned to be the second job $\varphi_{i2}^t = 2$ in the permutation φ_i^t, and so on. In other words, dimensions are sorted according to the smallest position values x_{ik}^t to construct the permutation of operations φ_i^t. Having determined the permutation of operations, the job repetition vector is determined. As an example, π_{11}^t value is obtained such that:

$$\pi_{11}^t = \left\lfloor \frac{\varphi_{11}^t - 1}{3} \right\rfloor + 1 = \left\lfloor \frac{6-1}{3} \right\rfloor + 1 = \lfloor 1.66 \rfloor + 1 = 1 + 1 = 2.$$

where the first part of the summation is floored to an integer number.

Finally, an active schedule is constructed from the job repetition vector π_i^t with the procedure given in Cheng et al. (1996). For example, consider the example in Table 2 and suppose that a job repetition vector is given as [2 1 2 2 1 3 1 3 3] where 1 stands for job J_1, 2 for job J_2, and 3 for job J_3. Since each job has three operations, it occurs three times in the job repetition vector. By scanning the job repetition vector from left to right, the k^{th} occurrence of a job refers to the k^{th} operation in the routing of the job. So the first 2 corresponds to the first operation of job J_2 which will be processed on machine M_1, the second 2 corresponds to the second operation of job J_2 which will be processed on machine M_3, and the third 2 corresponds to the third operation of job J_2 which will be processed on machine M_2. By doing so, the corresponding machine list can be obtained and an active schedule can be generated as shown in Figure 2.

Table 1. Solution representation

Dimension, k	1	2	3	4	5	6	7	8	9
x_{ik}^t	1.8	-0.99	3.01	0.72	-0.45	-2.25	5.3	4.8	1.9
φ_{ik}^t	6	2	5	4	1	9	3	8	7
π_{ik}^t	2	1	2	2	1	3	1	3	3

Table 2. An example of 3-job × 3-machine JSS problem

	Operations		
Jobs	M_1	M_2	M_3
	Processing Times		
J_1	3	3	3
J_2	2	3	4
J_3	3	2	1
	Routing (Machines)		
J_1	M_1	M_2	M_3
J_2	M_1	M_3	M_2
J_3	M_2	M_1	M_3

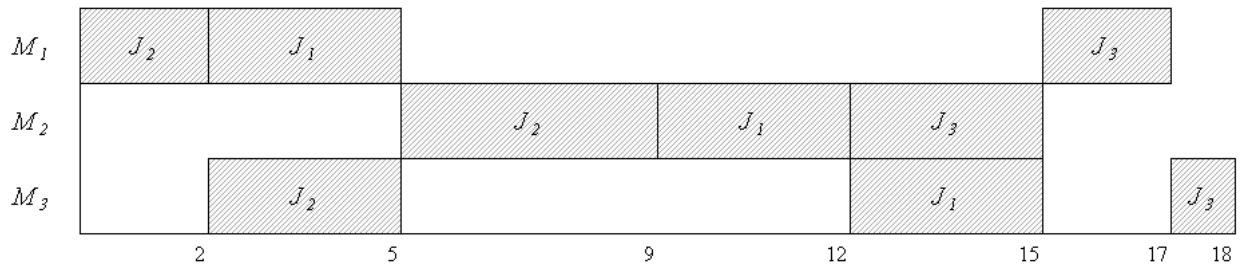

Figure 2. Active schedule generated from job repetition vector.

2.2 Initial population

The population of particles is constructed randomly and uniformly for the PSO algorithm of the JSS problem. The following formula is used to construct the continuous position values:

$$x_{ik}^0 = x_{\min} + (x_{\max} - x_{\min}) \times r,$$

where $x_{\min} = -4.0$, $x_{\max} = 4.0$ and r is a uniform random number between 0 and 1. Initial continuous velocities are generated by a similar formula as follows:

$v_{ik}^0 = v_{\min} + (v_{\max} - v_{\min}) \times r,$

where $v_{\min} = -4.0$, $v_{\max} = 4.0$, and r is a uniform random number between 0 and 1. Continuous velocity values are not restricted to a maximum and minimum range when updating the velocity. The population size is the number of dimensions. Since the objective is to minimize the makespan, the fitness function value, $f_i^t(\pi_i^t \leftarrow X_i^t)$, is the makespan value which is decoded from the job repetition vector for particle *i*. For simplicity, $f_i^t(\pi_i^t \leftarrow X_i^t)$ is denoted as f_i^t.

2.3 Computational flow

The complete computational flow of the PSO algorithm for the JSS problem is given below:

Step 1. Initialization

Set t = 0, NP = the number of dimensions, and $w^0 = 0.9$.

Generate NP particles randomly as explained before, $\{X_i^0, i = 1,2,...,NP\}$ where $X_i^0 = [x_{i1}^0, x_{i2}^0, ..., x_{i,nm}^0]$.

Generate the initial velocities for particle i randomly as explained before, $\{V_i^0, i = 1,2,...,NP\}$ where $V_i^0 = [v_{i1}^0, v_{i2}^0, ..., v_{i,nm}^0]$.Apply the SPV rule to find the permutation of operations $\varphi_i^0 = [\varphi_{i1}^0, \varphi_{i2}^0, ..., \varphi_{i,nm}^0]$ of particle X_i^0 for i = 1, 2, ..., NP.

Determine the job repetition vector $\pi_i^0 = [\pi_{i1}^0, \pi_{i2}^0, .., \pi_{i,nm}^0]$ of particle X_i^0 for i = 1, 2, ..., NP.

Evaluate each particle i in the swarm using the objective function f_i^0 for i = 1, 2, ..., NP.

For each particle i in the swarm, set the personal best to $P_i^0 = X_i^0$, where $P_i^0 = [p_{i1}^0 = x_{i1}^0, p_{i2}^0 = x_{i2}^0, ..., p_{i,nm}^0 = x_{i,nm}^0]$ together with its best fitness value, $f_i^{pb} = f_i^0$ for i = 1, 2, ..., NP.

Find the best fitness value among the whole swarm such that $f_l = \min\{f_i^0\}$ for i = 1, 2, ..., NP with its corresponding positions X_l^0. Set the global best to $G^0 = X_l^0$ such that $G^0 = [g_1 = x_{l,1}^0, g_2 = x_{l,2}^0, ..., g_{nm} = x_{l,nm}^0]$ with its fitness value $f^{gb} = f_l$.

Step 2. Update iteration counter

$t = t + 1$.

Step 3. Update inertia weight

$w^t = w^{t-1} \times \beta$ where β is the decrement factor which is a constant between (0, 1).

Step 4. Update velocity

$v_{ik}^t = w^{t-1} v_{ik}^{t-1} + c_1 r_1 (p_{ik}^{t-1} - x_{ik}^{t-1}) + c_2 r_2 (g_k^{t-1} - x_{ik}^{t-1})$

where c_1 and c_2 are social and cognitive parameters. r_1 and r_2 are uniform random numbers between (0, 1).

Step 5. Update position

$x_{ik}^t = x_{ik}^{t-1} + v_{ik}^t$.

Step 6. Find permutation of operations

Apply the SPV rule to find the permutation of operations $\varphi_i^t = [\varphi_{i1}^t, \varphi_{i2}^t, ..., \varphi_{i,nm}^t]$ for i = 1, 2, ..., NP.

Step 7. Find job repetition

Determine the job repetition vector, $\pi_i^t = [\pi_{i1}^t, \pi_{i2}^t, ..., \pi_{i,nm}^t]$ for i = 1, 2, ..., NP.

Step 8. Update the personal best

Each particle is evaluated by using the permutation to see if personal best will improve. That is, if $f_i^t < f_i^{pb}$ then the personal best is updated as $P_i^t = X_i^t$ and $f_i^{pb} = f_i^t$ for i = 1, 2, ..., NP.

Step 9. Update the global best

Find the minimum value of the personal best. That is, $f_l^t = \min\{f_i^{pb}\}$, i = 1, 2, ..., NP; $l \in$ {1, 2, ..., NP}.

If $f_l^t < f^{gb}$, then the global best is updated as $G^t = X_l^t$ and $f^{gb} = f_l^t$.

Step 10. Stopping criterion

If the CPU time exceeds the maximum CPU time limit, then stop; otherwise go to *step* 2.

3. DIFFERENTIAL EVOLUTION ALGORITHM FOR JSS PROBLEM

Solution representation and initial population in the DE algorithm are the same as the ones in the PSO algorithm except for omitting the velocity vector in the representation. Several variations of DE algorithms have been introduced in the literature. We follow the *DE/rand/1/bin* scheme of Storn and Price (1995) with the inclusion of the SPV rule in the algorithm. The pseudo code of the DE algorithm for the JSS problem is given in Figure 3.

Initialize parameters
Initialize target population
Find permutation of operations
Find job repetition
Evaluate
Do {
  Obtain mutant population
  Obtain trial population
  Find permutation of operations
  Find job repetition
  Evaluate trial population
  Do selection
  Apply local search
}*While (Not Termination)*

Figure 3. DE algorithm with local search for the JSS problem.

The basic elements of DE algorithm is summarized as follows:

Target individual: X_i^t denotes the i^{th} individual in the target population at generation t and is defined as

$X_i^t = [x_{i1}^t, x_{i2}^t, ..., x_{i,nm}^t]$, where x_{ik}^t is the dimension value of the i^{th} individual with respect to the k^{th} dimension (k = 1, 2, ..., nm).

Mutant individual: V_i^t denotes the i^{th} individual in the mutant population at generation t and is defined as $V_i^t = [v_{i1}^t, v_{i2}^t, ..., v_{i,nm}^t]$, where v_{ik}^t is the dimension value of the i^{th} individual with respect to the k^{th} dimension (k = 1, 2, ..., nm).

Trial individual: U_i^t denotes the i^{th} individual in the trial population at generation t and is defined as $U_i^t = [u_{i1}^t, u_{i2}^t, ..., u_{i,nm}^t]$, where u_{ik}^t is the dimension value of the i^{th} individual with respect to the k^{th} dimension (k = 1, 2, ..., nm).

Target population: X^t is the set of NP individuals in the target population at generation t, i.e., $X^t = [X_1^t, X_2^t, ..., X_{NP}^t]$.

Mutant population: V^t is the set of NP individuals in the mutant population at generation t, i.e., $V^t = [V_1^t, V_2^t, ..., V_{NP}^t]$.

Trial population: U^t is the set of NP individuals in the trial population at generation t, i.e., $U^t = [U_1^t, U_2^t, ..., U_{NP}^t]$.

Permutation of operations: A new variable φ_i^t, which is a permutation of the operations of jobs implied by the particle X_i^t is introduced. It can be described as $\varphi_i^t = [\varphi_{i1}^t, \varphi_{i2}^t, ..., \varphi_{i,nm}^t]$, where φ_{ik}^t is the assignment of operation k of the particle i in the permutation of operations at iteration t.

Job repetition: Another variable π_i^t, which is a repetition of jobs implied by the particle X_i^t is also introduced. It can be described as $\pi_i^t = [\pi_{i1}^t, \pi_{i2}^t, ..., \pi_{i,nm}^t]$, where π_{ik}^t is the assignment of job j of the particle i in the job repetition at iteration t. Note that each job repeats m times in the chromosome.

Mutant constant: $F \in (0, 2)$is a real number constant which affects the differential variation between two individuals.

Crossover constant: $CR \in (0, 1)$is a real number constant which affects the diversity of population for the next generation.

Fitness function: In a minimization problem, the objective function is $f_i(\pi_i^t \leftarrow X_i^t)$, where π_i^t is the corresponding job repetition vector of individual X_i^t.

The complete computational procedure of the DE algorithm for the JSS problem can be summarized as follows:

Step 1. Initialization

Set t = 0, NP = the number of dimensions.

Generate NP individuals randomly as explained before, $\{X_i^0, i = 1, 2, ..., NP\}$ where $X_i^0 = [x_{i1}^0, x_{i2}^0, ..., x_{i,nm}^0]$

Apply the SPV rule to find the permutation of operations $\varphi_i^0 = [\varphi_{i1}^0, \varphi_{i2}^0, ..., \varphi_{i,nm}^0]$ of particle X_i^0 for i = 1, 2, …, NP.

Determine the job repetition vector $\pi_i^0 = [\pi_{i1}^0, \pi_{i2}^0, ..., \pi_{i,nm}^0]$ of particle X_i^0 for i = 1, 2, …, NP.

Evaluate each individual i in the population using the objective function $f_i^0(\pi_i^o \leftarrow X_i^0)$ for i = 1, 2, …, NP.

Step 2. Update generation counter

$t = t + 1$.

Step 3. Generate mutant population

For each target individual, X_i^t, i = 1, 2, …, NP, at generation t, a mutant individual, $V_i^t = [v_{i1}^t, v_{i2}^t, ..., v_{i,nm}^t]$, is determined such that:

$$V_i^t = X_{a_i}^{t-1} + F \times (X_{b_i}^{t-1} - X_{c_i}^{t-1}),$$

where a_i, b_i, and c_i are three randomly chosen individuals from the population such that $(a_i \neq b_i \neq c_i)$.

Step 4. Generate trial population

Following the mutation phase, the crossover (recombination) operator is applied to obtain the trial population. For each mutant individual, $V_i^t = [v_{i1}^t, v_{i2}^t, .., v_{i,nm}^t]$, an integer random number between 1 and nm, i.e., $D_i \in$ (1, 2, …, nm), is chosen, and a trial individual, $U_i^t = [u_{i1}^t, u_{i2}^t, ..., u_{i,nm}^t]$ is generated such that:

$$u_{ik}^t = \begin{cases} v_{ik}^t, & if \quad r_{ik}^t \le CR \quad or \quad k = D_i, \\ x_{ik}^{t-1}, & otherwise, \end{cases} \tag{3}$$

where the index D refers to a randomly chosen dimension (k = 1, 2, ..., nm) used to ensure that at least one parameter of each trial individual U_i^t differs from its counterpart in the previous generation U_i^{t-1}, CR is a user-defined crossover constant in the range (0, 1), and r_{ik}^t is a uniform random number between 0 and 1. The trial individual is made up with some parameters of a mutant individual, or at least one of the parameters randomly selected, and some other parameters of the target individual.

Step 5. Find permutation of operations

Apply the SPV rule to find the permutation of operations $\varphi_i^t = [\varphi_{i1}^t, \varphi_{i2}^t, ..., \varphi_{i,nm}^t]$ for i = 1, 2, …, NP.

Step 6. Find job repetition

Determine the job repetition vector, $\pi_i^t = [\pi_{i1}^t, \pi_{i2}^t, ..., \pi_{i,nm}^t]$ for i = 1, 2, …, NP.

Step 7. Evaluate trial population

Evaluate the trial population using the objective function $f_i^t(\pi_i^t \leftarrow U_i^t)$ for $i = 1, 2, \ldots, NP$.

Step 8. Do selection

To decide whether or not the trial individual U_i^t should be a member of the target population for the next generation, it is compared to its counterpart target individual X_i^{t-1} at the previous generation. The selection is based on the survival of fitness among the trial population and target population such that:

$$X_i^t = \begin{cases} U_i^t, \ \textit{if } f(\pi_i^t \leftarrow U_i^t) \le f(\pi_i^{t-1} \leftarrow X_i^{t-1}), \\ X_i^{t-1}, \ \textit{otherwise}, \end{cases} \quad (4)$$

Step 9. Stopping criterion

If the CPU time exceeds the maximum CPU time limit, then stop; otherwise go to *step* 2.

4. NEIGHBORHOOD OF PSO AND DE ALGORITHMS

There might be two types of neighborhood structures that can be employed for the search. The first one is based on the neighbors of the positions whereas the second one is based on the neighbors of the job repetition vector. The second approach based on neighbors of job repetitions is used in this paper because of its efficiency. However, it violates the SPV rule and needs a repair mechanism. This approach is illustrated in Table 3 where $\pi_{i2}^t = 1$ and $\pi_{i6}^t = 3$ are interchanged. As shown in Table 3, the SPV rule is violated because the permutation of operations and job repetitions are the result of the particle's position values. After completing the local search, the particle should be repaired in order to satisfy the SPV rule. This is achieved by changing the values of position, and permutation of operations according to the SPV rule as shown in Table 4.

Table 3. Local search applied to job repetition before repairing

Dimension, k	1	2	3	4	5	6	7	8	9
x_{ik}^t	1.8	-0.99	3.01	0.72	-0.45	-2.25	5.3	4.8	1.9
φ_{ik}^t	6	2	5	4	1	9	3	8	7
π_{ik}^t	2	**1**	2	2	1	**3**	1	3	3
x_{ik}^t	1.8	-0.99	3.01	0.72	-0.45	-2.25	5.3	4.8	1.9
φ_{ik}^t	6	2	5	4	1	9	3	8	7
π_{ik}^t	2	**3**	2	2	1	**1**	1	3	3

Table 4. Local search applied to job repetition after repairing

Dimension, k	1	2	3	4	5	6	7	8	9
x_{ik}^t	1.8	-0.99	3.01	0.72	-0.45	-2.25	5.3	4.8	1.9
φ_{ik}^t	6	2	5	4	1	9	3	8	7
π_{ik}^t	2	**1**	2	2	1	**3**	1	3	3
x_{ik}^t	1.8	1.9	3.01	0.72	-0.45	-2.25	5.3	4.8	-0.99
φ_{ik}^t	6	9	5	4	1	2	3	8	7
π_{ik}^t	2	**3**	2	2	1	**1**	1	3	3

In other words, the values of positions and permutation of operations are interchanged in the particle. Sinc $\pi_{i2}^t = 1$ and $\pi_{i6}^t = 3$ are interchanged, their corresponding φ_{ik}^t and x_{ik}^t values are interchanged respectively to keep the particle consistent with the SPV rule. The advantage of this approach is due to the fact that the repair algorithm is only needed after evaluating all the neighbors in the job repetition vector.

The local search for the JSS problem is applied to the job repetition π^t of the global best solution at each iteration *t*. The performance of the local search algorithm depends on the choice of the neighborhood structure. Local search in this work is based on the *interchange+ insert* variant of the variable neighborhood search (VNS) method presented in Mladenovic and Hansen (1997). For the JSS problem, the following two neighborhood structures are employed:

Interchange two jobs between η^{th} and κ^{th} dimensions, $\eta \neq \kappa$ *(Interchange)*

Remove the job at the η^{th} dimension and insert it in the κ^{th} dimension $\eta \neq \kappa$ (*Insert*)

The pseudo code of the local search is given in Figure 4 where η and κ are the random integer numbers between 1 and *nm*. $s = \textit{insert}(s_0, \eta, \kappa)$ means removing the job from the η^{th} dimension in the job repetition vector s_0 and inserting it in the κ^{th} dimension in the job repetition vector s_0, resulting in a job repetition vector s. In case

of perturbing the job repetition vector s_0, two inserts and two interchanges were used to diversify the global best solution before applying the local search. This perturbation is important to direct the search towards the global optima since the global best solution remains the same after some iterations, probably at a local minimum. In addition, neutral moves are allowed in the VNS local search in order to restart the search from a different sequence with the same objective function value.

$s_0 = \pi^t$
$\eta = rnd(1, nm), \kappa = rnd(1, nm), \eta \neq \kappa$
$s = perturbation(s_0, \eta, \kappa)$
$loop = 0$
$Do\{$
  $kcount = 0$
  $\max\_method = 2$
  $Do\{$
    $\eta = rnd(1, nm), \kappa = rnd(1, nm), \eta \neq \kappa$
    $if\ (kcount == 0)\ s_1 = \text{int}\, exchange(s, \eta, \kappa)$
    $if\ (kcount == 1)\ s_1 = insert(s, \eta, \kappa)$
    $if\ (f(s_1) \leq f(s))\{$
      $kcount = 0$
      $s = s_1\ \}$
    $else\{kcount++\}$
  $\}While(kcount < \max\_method)$
  $loop++$
$\}While(loop < nm*(nm-1))$
$if(f(s) \leq f(\pi^t))\{$
  $\pi^t = s$
  $Repair(G^t)\}$

Figure 4. Pseudo code of VNS local search.

5. EXPERIMENTAL RESULTS

The proposed PSO and DE algorithms for the JSS problem are coded in C and run on an Intel Pentium IV 2.6 GHz PC with 256MB memory. The following parameters were used for the PSO and DE algorithms. The size of the population in both algorithms is the number of dimensions. The social and cognitive parameters were taken as $c_1 = c_2 = 2$ consistent with the literature. Initial inertia weight is set to $w^0 = 0.9$ and never decreased below 0.40. The decrement factor β is taken as 0.975. For the DE algorithm, mutant factor and crossover rate are taken as $F = 0.8$ and $CR = 0.9$ respectively.

First, different neighborhood structures such as *insert* + *interchange* and *interchange* + *insert* are hybridized in the PSO and DE algorithms for a set of known job shop benchmark problem instances from the literature. As shown in the experimental results in Tables 5 and 6, the *interchange* + *insert* structure of the VNS local search embedded in both algorithms generated lower relative errors, thus convincing us to use this neighborhood structure in further comparisons. Tables 5 and 6 summarize the statistics collected from the 20 replications for each instance. For each instance, its name and the best known upper bound (in parentheses) or optimal makespan (without parentheses) are given as reported by Jain and Meeran (1999). The third column gives the time limit to stop the algorithm. Since our comparisons are based on the recent work by Blum and Sampels (2004), the problem instances were run with different CPU time requirements to have a fair comparison. Owing to the fact that Blum and Sampels (2004) used a machine with 1.10 GHz, the maximum CPU times were restricted to the 11/26 of their CPU time limit because we employed a faster machine with 2.6 GHz. Similar correction factor of 300/2600 is used for the CPU time results of Murovec and Šuhel (2004) since they used a machine with Celeron 300 MHz. Experimental results are given in the next five columns for each algorithm. These are the best solution out of 20 runs, the average over 20 runs, the standard deviation of 20 runs, the average time (in seconds, $\bar{t}$) needed to reach the best objective function value in each run, and finally, the relative percent error based on the best solution out of 20 runs.

The performance measures in this paper were the average relative percent error and CPU time requirements. Computational efficiency was measured by the CPU time, and the solution quality was measured with the average relative percent error which is specifically defined as:

$$ARPE = \sum_{i=1}^{R} \left(\frac{(H_i - U_i) * 100}{U_i} \right) / R \qquad (5)$$

where H_i denotes the value of the makespan that the DE_{vns} or PSO_{vns}. algorithms generated, whereas U_i is the value of best known or optimal makespan provided in the literature, and R is the total number of problem instances.

Convergence graphs for the well-known problem instances of *ft*10 and *ft*20 are given in Figure 5 and 6. From the convergence graphs, it can be seen that Both PSO_{vns} and DE_{vns} algorithms were converged very quickly until 10 generations, very slowly thereafter. Since PSO_{vns} with the *interchange* + *insert* neighborhood structure has generated the lowest average relative percent error of 0.20% as shown in Tables 5 and 6, the results of PSO_{vns} with the *interchange* + *insert* neighborhood were used to compare with those recently published in the literature. However, DE_{vns} has performed well enough across the variety of problem instances in the literature.

The second comparison is based on the results of an ant colony optimization algorithm (ACO_GSS) by Blum and Sampels (2004). They reported the results for the 16 benchmark instances of the JSS problem and compared to their adaptation of the tabu search approach (TS_GSS) by Nowicki and Smutnicki (1996), the state-of-the-art

algorithms for the JSS problem. From Table 8, it can be seen that the hybrid PSO_{vns} algorithm outperforms the ACO_GSS algorithm in terms of the best, and average makespan as well as the average relative percent error generated since the statistics are 994.00, 1002.23 and 0.28 for the hybrid PSO_{vns} whereas it is 1000.50, 1007.81, and 1.02 for ACO_GSS. However, the ACO_GSS algorithm is more robust than the hybrid PSO_{vns} algorithm due to the lower standard deviations. As explained before, the CPU time limits were fixed to 11/26 of those reported in Blum and Sampels (2004) because of the faster machine we used. However, the average time to reach the best solution in each run is still smaller for PSO_{vns} than ACO_GSS. In addition to above, ACO_GSS is able to find only the 6 best known or optimal solutions whereas the hybrid PSO_{vns} algorithm is able to find the 10 best known or optimal solutions among 16 instances reported.

Table 5. Performance comparison of PSO_{vns} and DE_{vns} with *interchange + insert version*

			PSO_{vns}					DE_{vns}				
Instance	Best Known	Time limit	Best	Avg	$\sqrt{\sigma^2}$	$\bar{t}$	RPE	Best	Avg	$\sqrt{\sigma^2}$	$\bar{t}$	RPE
*abz*05	**1234**	76.15	**1234**	1236.25	2.31	17.36	0.00	**1234**	1235.80	2.09	25.66	0.00
*abz*06	**943**	76.15	**943**	943.00	0.00	12.49	0.00	**943**	943.00	0.00	10.70	0.00
*abz*07	**(656)**	761.54	659	670.10	5.75	573.59	0.46	666	672.70	3.84	444.09	1.52
*abz*08	**(665)**	761.54	674	682.30	5.52	442.87	1.35	674	681.90	6.13	476.71	1.35
*abz*09	**(679)**	761.54	688	697.55	5.79	424.79	1.33	682	697.90	8.80	479.08	0.44
*ft*10	**930**	76.15	**930**	938.45	9.71	36.14	0.00	**930**	935.80	10.91	26.38	0.00
*ft*20	**1165**	76.15	**1165**	1175.25	5.30	17.68	0.00	**1165**	1172.15	6.64	20.34	0.00
*orb*01	**1059**	76.15	**1059**	1076.05	10.58	25.29	0.00	1064	1078.50	11.02	34.90	0.47
*orb*02	**888**	76.15	889	889.45	1.79	10.15	0.11	889	891.40	3.76	14.73	0.11
*orb*03	**1005**	76.15	**1005**	1034.55	22.95	31.37	0.00	**1005**	1035.70	21.99	36.93	0.00
*orb*04	**1005**	76.15	**1005**	1011.30	6.42	41.63	0.00	**1005**	1012.85	6.44	26.89	0.00
*orb*05	**887**	76.15	**887**	892.45	4.81	27.14	0.00	889	892.95	7.76	30.20	0.23
*orb*06	**1010**	76.15	1013	1018.80	5.73	26.49	0.30	**1010**	1020.45	6.27	44.22	0.00
*orb*07	**397**	76.15	**397**	398.60	2.56	15.03	0.00	**397**	399.00	2.60	13.63	0.00
*orb*08	**899**	76.15	**899**	913.40	14.19	36.07	0.00	**899**	913.25	12.70	27.09	0.00
*orb*09	**934**	76.15	**934**	939.45	4.27	8.08	0.00	**934**	940.25	3.65	11.79	0.00
*orb*10	**944**	76.15	**944**	944.00	0.00	16.43	0.00	**944**	944.35	1.57	17.06	0.00
*la*16	**945**	76.15	**945**	948.50	9.08	20.63	0.00	**945**	946.85	6.88	24.45	0.00
*la*19	**842**	76.15	**842**	844.00	3.78	19.09	0.00	**842**	844.95	4.96	29.63	0.00
*la*21	**1046**	380.77	1047	1053.80	6.01	146.18	0.10	**1047**	1055.40	7.39	92.50	0.10
*la*22	**927**	380.77	**927**	930.55	2.95	107.16	0.00	**927**	930.10	3.42	109.91	0.00
*la*24	**935**	380.77	**935**	939.70	3.63	182.19	0.00	938	940.70	5.96	169.17	0.32
*la*25	**977**	380.77	**977**	981.45	4.74	143.12	0.00	**977**	981.35	2.70	129.38	0.00
*la*27	**1235**	761.54	**1235**	1248.10	10.09	359.36	0.00	**1235**	1250.45	11.20	367.22	0.00
*la*28	**1216**	761.54	**1216**	1216.25	0.64	180.44	0.00	**1216**	1216.45	1.57	192.71	0.00
*la*29	**1152**	761.54	1164	1176.70	10.55	353.93	1.04	1163	1172.30	6.35	484.87	0.95
*la*36	**1268**	761.54	**1268**	1279.30	6.98	464.49	0.00	**1268**	1275.25	5.93	443.62	0.00
*la*37	**1397**	761.54	**1397**	1410.90	7.74	316.75	0.00	**1397**	1414.50	7.41	318.38	0.00
*la*38	**1196**	761.54	**1196**	1212.50	14.96	415.13	0.00	**1196**	1206.20	5.46	474.19	0.00
*la*39	**1233**	761.54	**1233**	1240.00	4.10	387.69	0.00	**1233**	1238.10	5.75	367.40	0.00
*la*40	**1222**	761.54	1224	1227.60	3.80	378.76	0.16	1224	1228.10	5.57	391.80	0.16
*yn*01	**(888)**	1523.08	893	901.15	6.56	818.47	0.56	894	902.75	5.58	870.18	0.68
*yn*02	**(909)**	1523.08	910	925.85	6.43	962.93	0.11	917	925.95	3.69	1089.17	0.88
*yn*03	**(893)**	1523.08	902	908.40	5.23	1007.06	1.01	895	906.85	6.93	1086.82	0.22
*yn*04	**(968)**	1523.08	973	987.05	8.87	1015.96	0.52	980	990.55	7.98	891.95	1.24
Mean					**6.40**	**258.34**	**0.20**			**6.31**	**264.96**	**0.25**

The third comparison was for the TS_GSS algorithm, which has already been shown in Blum and Sampels (2004), outperforming the ACO_GSS algorithm. Table 9 presents the comparison of the hybrid PSO_{vns} to TS_GSS. In terms of the best makespan and relative percent error generated, the hybrid PSO_{vns} algorithm generated slightly better results than the TS_GSS algorithm because the mean best and RPE for PSO_{vns} is 994.00 and 0.28% respectively whereas it is 994.63 and 0.32% for TS_GSS. However, the TS_GSS algorithm generated much more robust results due to the much lower standard deviations and it was slightly faster than PSO_{vns}. To sum up, the hybrid PSO_{vns} algorithm has produced smaller best makespan and lower relative percent errors, but the TS_GSS and ACO_GSS algorithms, on the other hand, are more robust than the hybrid PSO_{vns} algorithm. Note that the tabu search adapted

in Blum and Sampels (2004) is one of the state-of-the-art algorithms for the JSS problem. These results can be interpreted in another way. Blum and Sampels (2004) developed an extension of neighborhood structure presented by Nowicki and Smutnicki (1996). This extension of neighborhood structure is used both in their ant colony algorithm as a steepest descent local search and in their adaptation of tabu search as a move structure. From Table 8, it can concluded that the simple VNS structure hybridized with the PSO algorithm was very effective in finding better or competitive results for a set of well known benchmark instances than those generated by their extension of neighborhood structure presented by Nowicki and Smutnicki (1996) hybridized both with the ACO and TS algorithms. This is documented by another fact that TS_GSS is able to find 9 best known or optimal solutions out of 16 problem instances whereas PSO_{vns} is able to find 10 best known or optimal solutions out of 16 problem instances. The last comparison is due to the most recent study (GA_TS) by Murovec and Šuhel (2004) in which they developed a genetic algorithm employing a tabu search with a repairing technique as a local search. Upper bounds of three instances are improved by Murovec and Šuhel (2004) where five hundred replications for each instance are conducted. From Table 10, it is clear that the GA_TS algorithm is superior to the PSO_{vns} algorithm in terms of all the performance measure. However, it is also superior to dESA, ACO_GSS and TS_GSS algorithms too. Murovec and Šuhel (2004) clearly reported the best results so far in the literature.

Table 6. Performance comparison of PSO and DE with *insert + interchange version*

			PSO_{vns}					DE_{vns}				
Instance	Best Known	Time limit	Best	Avg	$\sqrt{\sigma^2}$	$\bar{t}$	RPE	Best	Avg	$\sqrt{\sigma^2}$	$\bar{t}$	RPE
*abz*05	**1234**	76.15	**1234**	1236.3	2.13	30.70	0.00	**1234**	1235.15	1.95	17.88	0.00
abz 06	**943**	76.15	**943**	943.00	0.00	11.85	0.00	**943**	943.00	0.00	14.93	0.00
abz 07	**(656)**	761.54	666	672.40	4.45	441.21	1.52	661	670.60	4.86	521.31	0.76
abz 08	**(665)**	761.54	675	680.65	5.02	476.96	1.50	671	681.70	6.71	445.33	0.90
abz 09	**(679)**	761.54	691	697.55	4.35	479.66	1.77	685	694.60	5.56	539.82	0.88
*ft*10	**930**	76.15	**930**	942.25	12.77	27.83	0.00	**930**	939.65	11.25	28.43	0.00
*ft*20	**1165**	76.15	**1165**	1173.30	6.28	26.42	0.00	**1165**	1173.95	6.05	19.58	0.00
*orb*01	**1059**	76.15	1064	1074.40	8.48	38.24	0.47	**1059**	1075.30	13.64	39.60	0.00
orb 02	**888**	76.15	**888**	889.90	2.51	21.89	0.00	889	890.60	3.28	15.45	0.11
orb 03	**1005**	76.15	1017	1038.15	13.87	44.39	1.19	**1005**	1034.45	24.67	33.52	0.00
orb 04	**1005**	76.15	**1005**	10120	4.69	37.00	0.00	**1005**	1012.90	5.74	26.70	0.00
orb 05	**887**	76.15	889	892.20	3.97	36.01	0.23	**887**	896.50	11.83	19.77	0.00
orb 06	**1010**	76.15	1013	1020.20	6.40	31.95	0.30	**1010**	1017.70	6.43	34.62	0.00
orb 07	**397**	76.15	**397**	399.40	3.02	16.66	0.00	**397**	398.70	2.70	12.76	0.00
orb 08	**899**	76.15	**899**	910.85	13.28	37.08	0.00	**899**	915.80	18.44	27.48	0.00
orb 09	**934**	76.15	**934**	940.60	3.68	12.91	0.00	**934**	939.05	4.10	17.81	0.00
orb 10	**944**	76.15	**944**	944.00	0.00	20.92	0.00	**944**	944.00	0.00	16.55	0.00
*la*16	**945**	76.15	**945**	955.05	13.97	17.54	0.00	**945**	948.45	9.10	23.29	0.00
la 19	**842**	76.15	**842**	843.85	4.59	30.26	0.00	**842**	843.30	3.26	31.89	0.00
la 21	**1046**	380.77	1047	1054.55	5.06	85.09	0.10	1047	1055.30	8.23	172.77	0.10
la 22	**927**	380.77	**927**	928.90	2.51	116.66	0.00	**927**	929.50	2.70	105.47	0.00
la 24	**935**	380.77	**935**	939.35	2.72	216.34	0.00	938	939.95	2.19	165.85	0.32
la 25	**977**	380.77	**977**	983.15	4.83	113.60	0.00	**977**	982.55	4.11	114.69	0.00
la 27	**1235**	761.54	1236	1249.65	9.37	456.01	0.08	**1235**	1244.40	8.85	412.48	0.00
la 28	**1216**	761.54	**1216**	1216.00	0.00	222.18	0.00	**1216**	1216.25	0.55	246.71	0.00
la 29	**1152**	761.54	1164	1172.90	5.24	442.26	1.04	1163	1171.60	6.82	519.88	0.95
la 36	**1268**	761.54	**1268**	1278.35	7.00	454.95	0.00	**1268**	1275.00	4.74	417.70	0.00
la 37	**1397**	761.54	**1397**	1410.75	8.13	439.40	0.00	**1397**	1409.45	8.15	433.94	0.00
la 38	**1196**	761.54	**1196**	1206.25	7.06	441.11	0.00	1207	1214.70	14.96	365.34	0.92
la 39	**1233**	761.54	**1233**	1238.90	5.82	361.96	0.00	**1233**	1240.25	5.31	386.02	0.00
la 40	**1222**	761.54	1224	1229.40	7.34	398.29	0.16	1224	1227.35	3.53	422.99	0.16
*yn*01	**(888)**	1523.08	892	900.85	5.67	939.82	0.45	895	902.75	4.71	904.59	0.79
yn 02	**(909)**	1523.08	911	923.85	6.21	1002.95	0.22	917	924.20	5.66	998.22	0.88
yn 03	**(893)**	1523.08	897	908.85	6.56	1004.19	0.45	898	905.95	5.27	1090.40	0.56
yn 04	**(968)**	1523.08	975	987.75	6.15	1042.97	0.72	981	989.85	4.96	1075.22	1.34
Mean					**5.80**	**273.64**	**0.29**			**6.58**	**277.69**	**0.25**

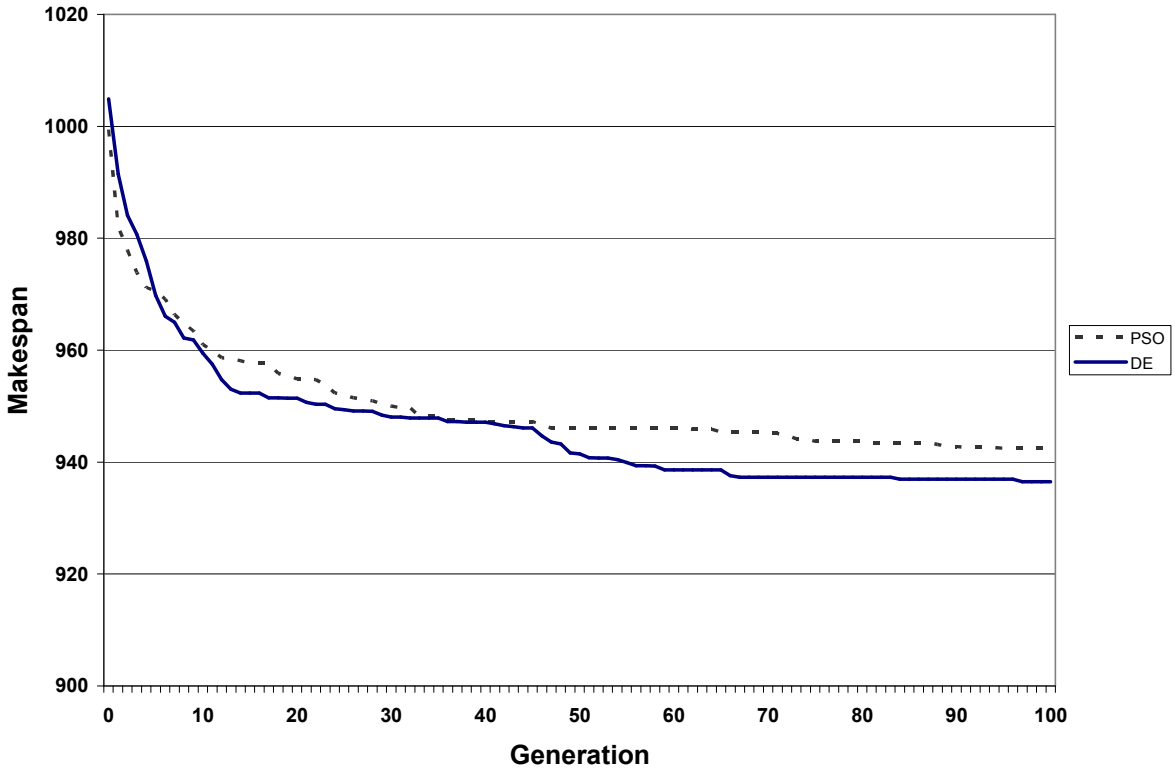

Figure 5. Convergence graph for the *ft*10 in terms of average relative percent deviation.

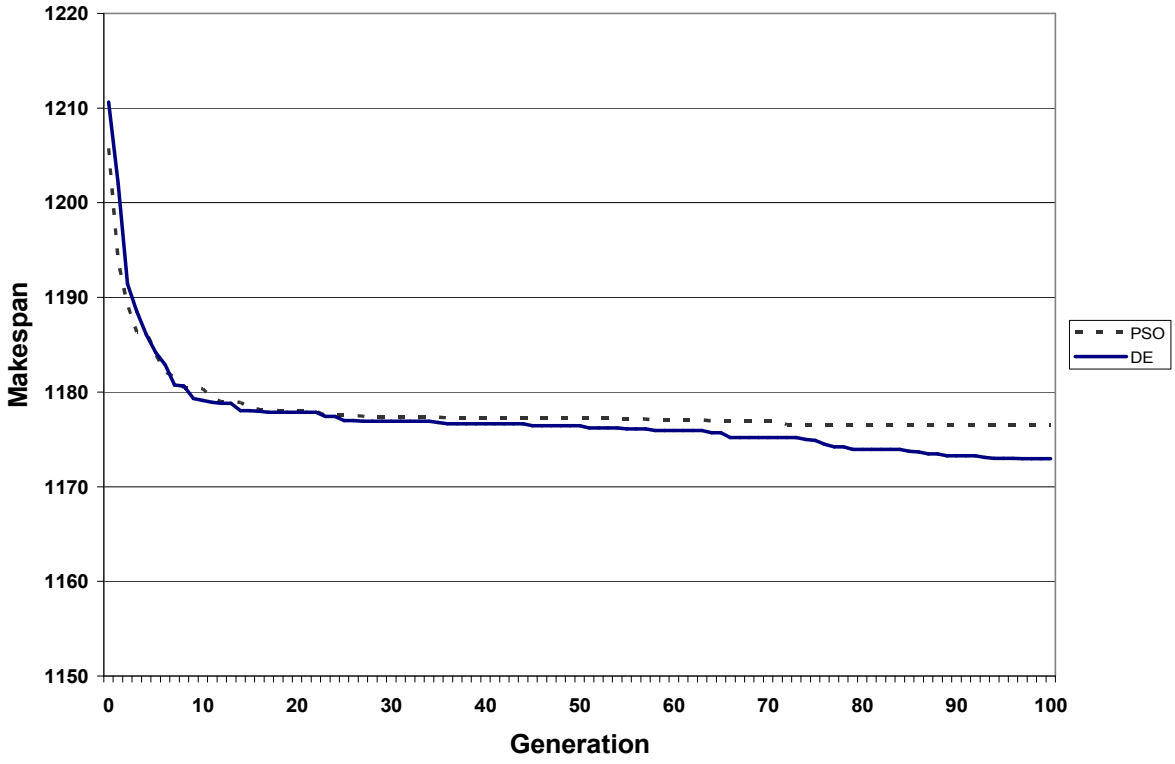

Figure 6. Convergence graph for the *ft*20 in terms of average relative percent deviation.

Table 7. Performance comparison of PSO_{vns} and dESA

			PSO_{vns}					dESA				
Instance	Best known	Time limit	Best	Avg	$\sqrt{\sigma^2}$	$\bar{t}$	RPE	Best	Avg	$\sqrt{\sigma^2}$	$\bar{t}$	RPE
*abz*07	**(656)**	761.54	659	670.10	5.75	573.59	0.46	672	682.40	8.11	NA	2.44
abz 08	**(665)**	761.54	674	682.30	5.52	442.87	1.35	681	691.50	7.34	NA	2.41
abz 09	**(679)**	761.54	688	697.55	5.79	424.79	1.33	699	706.60	6.06	NA	2.95
*la*21	1046	380.77	1047	1053.80	6.01	146.18	0.10	**1046**	1049.40	4.17	NA	0.00
la 24	935	380.77	**935**	939.70	3.63	182.19	0.00	938	940.80	2.97	NA	0.32
la 25	977	380.77	**977**	981.45	4.74	143.12	0.00	**977**	982.00	5.06	NA	0.00
la 27	1235	761.54	**1235**	1248.10	10.09	359.36	0.00	1240	1242.00	4.52	NA	0.41
la 29	1152	761.54	1164	1176.70	10.55	353.93	1.04	1176	1179.00	9.81	NA	2.08
la 38	1196	761.54	**1196**	1212.50	14.96	415.13	0.00	1201	1208.40	5.02	NA	0.42
*la*40	1222	761.54	1224	1227.60	3.80	378.76	0.16	1228	1232.40	5.84	NA	0.49
Mean			**979.90**	**988.98**	**7.08**	**341.99**	**0.44**	**985.80**	**991.45**	**5.89**	**NA**	**1.15**

Table 8. Performance comparison of PSOvns and ACO_GSS

			PSO_{vns}					ACO_GSS				
Instance	Best known	Time limit	Best	Avg	$\sqrt{\sigma^2}$	$\bar{t}$	RPE	Best	Avg	$\sqrt{\sigma^2}$	$\bar{t}$	RPE
*abz*07	**(656)**	761.54	659	670.10	5.75	573.59	0.46	674	681.20	3.16	405.63	2.74
*abz*08	**(665)**	761.54	674	682.30	5.52	442.87	1.35	689	697.05	3.24	462.53	3.61
*abz*09	**(679)**	761.54	688	697.55	5.79	424.79	1.33	702	709.35	4.16	448.13	3.39
*la*21	1046	380.77	1047	1053.80	6.01	146.18	0.10	1047	1053.25	3.51	194.73	0.10
*la*24	935	380.77	**935**	939.70	3.63	182.19	0.00	944	948.10	3.39	153.84	0.96
*la*25	977	380.77	**977**	981.45	4.74	143.12	0.00	**977**	981.45	2.98	378.49	0.00
*la*27	1235	761.54	**1235**	1248.10	10.09	359.36	0.00	1243	1255.50	5.90	436.51	0.65
*la*29	1152	761.54	1164	1176.70	10.55	353.93	1.04	1168	1186.75	8.15	459.03	1.39
*la*38	1196	761.54	**1196**	1212.50	14.96	415.13	0.00	1227	1235.45	4.17	392.62	2.59
*la*40	1222	761.54	1224	1227.60	3.80	378.76	0.16	1228	1234.55	5.92	436.23	0.49
*ft*10	930	76.15	**930**	938.45	9.71	36.14	0.00	**930**	938.90	7.61	39.55	0.00
*ft*20	1165	76.15	**1165**	1175.25	5.30	17.68	0.00	**1165**	1168.55	5.11	37.35	0.00
*orb*08	899	76.15	**899**	913.40	14.19	36.07	0.00	**899**	914.65	6.87	37.34	0.00
*orb*09	934	76.15	**934**	939.45	4.27	8.08	0.00	**934**	935.15	2.92	34.06	0.00
*abz*05	1234	76.15	**1234**	1236.25	2.31	17.36	0.00	**1234**	1237.20	1.36	14.46	0.00
*abz*06	943	76.15	**943**	943.00	0.00	12.49	0.00	947	947.80	0.41	6.50	0.42
Mean			**994.00**	**1002.23**	**6.66**	**221.73**	**0.28**	**1000.50**	**1007.81**	**4.30**	**246.06**	**1.02**

6. CONCLUSIONS

In this paper, we present PSO and DE algorithms for the JSS problem with the makespan criterion. The applications of PSO and DE on combinatorial optimization problems are still considered limited, but the advantages of PSO and DE algorithms such as structural simplicity, accessibility to practical applications, ease of implementation, speed to get the solutions, and robustness are already shown in the literature. However, the major obstacle of successfully applying PSO and DE algorithms to combinatorial optimization problems is due to their continuous nature. To remedy this drawback, the SPV rule presented in Tasgetiren et al. (2004a, b, c, d) is used in both algorithms to convert continuous position values to discrete job permutations. Both algorithms are also hybridized with an efficient local search method based on a VNS method in order to improve the solution quality.

To the best of our knowledge, these are the first reported applications of the PSO and DE algorithms to the JSS problem with either better or competitive results to the well known approaches in the literature. The SPV rule can be employed to enable the continuous PSO and DE algorithms to be applied to all classes of sequencing and scheduling problems. It is hoped that the PSO and DE algorithms with the proposed SPV rule will enrich the PSO and DE literatures applied to the combinatorial optimization problems in the future.

As summarized in Table 11, it has been shown that the hybrid PSO_{vns} algorithm generated better results than the dESA algorithm. It has also been shown that the hybrid PSO_{vns} algorithm generated better results than the ACO_GSS algorithm, and was very competitive to TS_GSS algorithm. However, the hybrid PSO_{vns} was not so robust in comparison to dESA, ACO_GSS and TS_GSS.

The hybrid PSO_{vns} algorithm was not competitive to the GA_TS algorithm which generated best reported results so far in the literature. However, both PSO_{vns} and DE_{vns} algorithms solved very hard instances collected from the OR library with 0.20% and 0.25% deviations from the best known or optimal solutions reported in the literature.

Table 9. Performance comparison of PSO and TS_GSS

			PSO_{vns}					TS_GSS				
Instance	Best known	Time limit	Best	Avg	$\sqrt{\sigma^2}$	$\bar{t}$	RPE	Best	Avg	$\sqrt{\sigma^2}$	$\bar{t}$	RPE
*abz*07	**(656)**	761.54	659	670.10	5.75	573.59	0.46	666	668.45	1.47	350.30	1.52
*abz*08	**(665)**	761.54	674	682.30	5.52	442.87	1.35	673	679.95	3.17	335.93	0.60
*abz*09	**(679)**	761.54	688	697.55	5.79	424.79	1.33	688	692.20	2.61	267.42	1.33
*la*21	1046	380.77	1047	1053.80	6.01	146.18	0.10	1047	1049.25	2.05	155.74	0.10
*la*24	935	380.77	**935**	939.70	3.63	182.19	0.00	939	942.30	1.38	131.98	0.43
*la*25	977	380.77	**977**	981.45	4.74	143.12	0.00	**977**	977.30	0.47	286.35	0.00
*la*27	1235	761.54	**1235**	1248.10	10.09	359.36	0.00	**1235**	1241.15	3.69	379.06	0.00
*la*29	1152	761.54	1164	1176.70	10.55	353.93	1.04	1164	1168.10	2.17	347.78	1.04
*la*38	1196	761.54	**1196**	1212.50	14.96	415.13	0.00	**1196**	1201.40	1.85	366.88	0.00
*la*40	1222	761.54	1224	1227.60	3.80	378.76	0.16	1224	1228.35	2.52	300.94	0.16
*ft*10	930	76.15	**930**	938.45	9.71	36.14	0.00	**930**	931.90	3.32	28.10	0.00
*ft*20	1165	76.15	**1165**	1175.25	5.30	17.68	0.00	**1165**	1165.00	0.00	9.52	0.00
*orb*08	899	76.15	**899**	913.40	14.19	36.07	0.00	**899**	910.75	6.33	29.78	0.00
*orb*09	934	76.15	**934**	939.45	4.27	8.08	0.00	**934**	934.00	0.00	11.77	0.00
*abz*05	1234	76.15	**1234**	1236.25	2.31	17.36	0.00	**1234**	1236.90	1.37	23.05	0.00
*abz*06	943	76.15	**943**	943.00	0.00	12.49	0.00	**943**	943.70	0.98	25.99	0.00
Mean			**994.0**	**1002.23**	**6.66**	**221.73**	**0.28**	**994.63**	**998.17**	**2.09**	**190.66**	**0.32**

Table 10. Performance comparison of PSO_{vns} and GA_TS

			PSO_{vns}					GA_TS				
Instance	Best known	Time limit	Best	Avg	Worst	$\bar{t}$	RPE	Best	Avg	Worst	$\bar{t}$	RPE
*abz*05	**1234**	76.15	**1234**	1236.25	1239.00	17.36	0.00	**1234**	1234.90	1238	0.61	0.00
*abz*06	**943**	76.15	**943**	943.00	943.00	12.49	0.00	**943**	943.00	943	0.28	0.00
*abz*07	**(656)**	761.54	659	670.10	682.00	573.59	0.46	658	666.40	674	77.75	0.30
*abz*08	**(665)**	761.54	674	682.30	691.00	442.87	1.35	669	674.30	685	104.88	0.60
*abz*09	**(679)**	761.54	688	697.55	708.00	424.79	1.33	**678**	687.50	701	93.39	-0.15
*ft*10	**930**	76.15	**930**	938.45	967.00	36.14	0.00	**930**	931.60	944	1.15	0.00
*ft*20	**1165**	76.15	**1165**	1175.25	1178.00	17.68	0.00	**1165**	1165.20	1173	1.11	0.00
*orb*01	**1059**	76.15	**1059**	1076.05	1099.00	25.29	0.00	**1059**	1062.40	1077	1.01	0.00
*orb*02	**888**	76.15	889	889.45	897.00	10.15	0.11	**888**	888.60	890	0.54	0.00
*orb*03	**1005**	76.15	**1005**	1034.55	1077.00	31.37	0.00	**1005**	1012.10	1035	1.44	0.00
*orb*04	**1005**	76.15	**1005**	1011.30	1023.00	41.63	0.00	**1005**	1008.10	1013	0.68	0.00
*orb*05	**887**	76.15	**887**	892.45	904.00	27.14	0.00	**887**	888.30	891	0.91	0.00
*orb*06	**1010**	76.15	1013	1018.80	1031.00	26.49	0.30	**1010**	1012.80	1023	1.39	0.00
*orb*07	**397**	76.15	**397**	398.60	403.00	15.03	0.00	**397**	397.00	397	0.39	0.00
*orb*08	**899**	76.15	**899**	913.40	944.00	36.07	0.00	**899**	902.40	927	1.6	0.00
*orb*09	**934**	76.15	**934**	939.45	943.00	8.08	0.00	**934**	934.70	943	0.74	0.00
*orb*10	**944**	76.15	**944**	944.00	944.00	16.43	0.00	**944**	944.00	944	0.46	0.00
*la*16	**945**	76.15	**945**	948.50	976.00	20.63	0.00	**945**	945.00	945	0.33	0.00
*la*19	**842**	76.15	**842**	844.00	852.00	19.09	0.00	**842**	842.10	848	0.38	0.00
*la*21	**1046**	380.77	1047	1053.80	1071.00	146.18	0.10	**1046**	1048.70	1055	3.07	0.00
*la*22	**927**	380.77	**927**	930.55	935.00	107.16	0.00	**927**	927.70	935	3.34	0.00
*la*24	**935**	380.77	**935**	939.70	950.00	182.19	0.00	**935**	938.20	943	3.07	0.00
*la*25	**977**	380.77	**977**	981.45	998.00	143.12	0.00	**977**	978.20	984	3.71	0.00
*la*27	**1235**	761.54	**1235**	1248.10	1264.00	359.36	0.00	**1235**	1236.80	1256	16.07	0.00
*la*28	**1216**	761.54	**1216**	1216.25	1218.00	180.44	0.00	**1216**	1216.00	1216	2.4	0.00
*la*29	**1152**	761.54	1164	1176.70	1205.00	353.93	1.04	1153	1165.20	1178	16.3	0.09
*la*36	**1268**	761.54	**1268**	1279.30	1291.00	464.49	0.00	**1268**	1268.30	1278	15.28	0.00
*la*37	**1397**	761.54	**1397**	1410.90	1421.00	316.75	0.00	**1397**	1402.40	1418	30.2	0.00
*la*38	**1196**	761.54	**1196**	1212.50	1248.00	415.13	0.00	**1196**	1202.30	1212	24.51	0.00
*la*39	**1233**	761.54	**1233**	1240.00	1248.00	387.69	0.00	**1233**	1235.50	1248	25.18	0.00
*la*40	**1222**	761.54	1224	1227.60	1237.00	378.76	0.16	**1222**	1226.00	1234	22.9	0.00
*yn*01	**(888)**	1523.08	893	901.15	914.00	818.47	0.56	**886**	896.00	905	220.78	-0.23
*yn*02	**(909)**	1523.08	910	925.85	937.00	962.93	0.11	**907**	915.70	927	242.28	-0.22
*yn*03	**(893)**	1523.08	902	908.40	919.00	1007.06	1.01	**895**	900.60	909	228.23	0.22
*yn*04	**(968)**	1523.08	973	987.05	1010.00	1015.96	0.52	969	978.50	991	263.36	0.10
Mean			**988.83**	**996.94**	**1010.49**	**258.34**	**0.20**	**987.26**	**990.76**	**999.43**	**40.28**	**0.02**

Table 11. Summary of comparisons

Problems	Best Known	PSO_{vns}	DE_{vns}	dESA	ACO_GSS	TS_GSS	GA_TS
*abz*07	656	0.46	1.52	2.44	2.74	1.52	0.30
*abz*08	665	1.35	1.35	2.41	3.61	0.60	0.60
*abz*09	679	1.33	0.44	2.95	3.39	1.33	-0.15
*la*21	1046	0.10	0.10	0.00	0.10	0.10	0.00
*la*24	935	0.00	0.32	0.32	0.96	0.43	0.00
*la*25	977	0.00	0.00	0.00	0.00	0.00	0.00
*la*27	1235	0.00	0.00	0.41	0.65	0.00	0.00
*la*29	1152	1.04	0.95	2.08	1.39	1.04	0.09
*la*38	1196	0.00	0.00	0.42	2.59	0.00	0.00
*la*40	1222	0.16	0.16	0.49	0.49	0.16	0.00
Mean		**0.44**	**0.48**	**1.15**	**1.59**	**0.52**	**0.08**

To summarize, the results presented in this work are very encouraging and promising for the applications of the PSO and DE algorithms to job shop scheduling problems, and hence to the other scheduling problems. It should be noted that the success was mainly due to the use of VNS local search improving the solution quality together with the neutral moves allowed.

For future work, the SPV rule can be used in both algorithms to solve other combinatorial optimization problems requiring permutation representation. It should also be pointed out that VNS is so time consuming due to the nature of changing the neighborhood during the search. This is one of our future research directions such that the iterated local search (ILS) with an insert neighborhood could be used to further improve the performance of the PSO algorithm. In addition, a deterministic version (VND) of the VNS local search with the first improvement pivoting rule could be used to improve the performance of the current algorithm

REFERENCES

1. Aarts, E. and Lenstra., J.K. (1997). *Local Search and Combinatorial Optimization*, Wiley, New York.
2. Abido, M.A. (2002). Optimal power flow using particle swarm optimization. *Electrical Power and Energy Systems*, 24: 563-571.
3. Adams, J., Balas, E., and Zawack, D. (1988). The Shifting Bottleneck Procedure for Job Shop Scheduling. *Management Science*, 34: 391-401.
4. Aiex, R.M., Binato, S., and Resende, M.G.C. (2003). Parallel GRASP with path-relinking for job shop Scheduling. *Parallel Computing*, 29: 393-430.
5. Applegate, D. and Cook, W. (1991). A computational study of job-shop scheduling. *ORSA Journal on Computing*, 3(2): 149-156.
6. Aydin, M.E. and Fogarty, T.C. (2004) A distributed evolutionary simulated annealing algorithm for combinatorial optimisation problems. *Journal of Heuristics*, 10: 269-292.
7. Balas, E. and Vazacopoulos, A. (1998). Guided local search with shifting bottleneck for job shop Scheduling. *Management Science*, 44: 262-275.
8. Bean, J.C. (1994). Genetic algorithm and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2): 154-160.
9. Bierwith, C. (1995). A generalized permutation approach to job shop scheduling with genetic algorithms. *OR Spektrum*, 17: 87-92.
10. Blazewicz, J., Domschke, W., and Pesch, E. (1996). The job shop scheduling problem: conventional and new solution techniques. *European Journal of Operations Research*, 93: 1-33.
11. Blum, C. and Sampels, M. (2004). An ant colony optimization algorithm for shop scheduling problems. *Journal of Mathematical Modelling and Algorithms*, 3: 285-308.
12. Brandstatter, B. and Baumgartner, U. (2002). Particle swarm optimization—mass-spring system analogon. *IEEE Transactions on Magnetics* 38: 997-1000.
13. Brinkkötter, W. and Brucker, P. (2001). Solving open benchmark problems for the job shop problem. *Journal of Scheduling* 4: 53-64.
14. Carlier, J. and Pison, E. (1989). An algorithm for solving the job-shop problem. *Management Science*, 35: 164-176.
15. Cheng, R., Gen, M., and Tsujimura, Y. (1996). A tutorial survey of job shop scheduling problems using genetic algorithms-I. Representation. *Journal of Computers and Industrial Engineering*, 30(4): 983-997.
16. Colorni, A., Dorigo, M., Maniezzo, V., and Trubian, M. (1994). Ant system for job-shop scheduling. *belgian Journal of Operations Research, Statistics and Computer Science (JORBEL)*, 34(1): 39-53.
17. Dell'Amico, M. and Trubian, M. (1993). Applying tabu-search to the job-shop scheduling problem. *Annals of Operations Research*, 4: 231-252.
18. Demirkol, E., Mehta, S., and Uzsoy, R. (1998). Benchmarks for shop scheduling problems. *European Journal of Operational Research*, 109: 137-141.
19. Dorndorf, U. and Pesch, E. (1995). Evolution based learning in a job shop scheduling environment. *Computers and Operations Research*, 22: 25-40.
20. Dorndorf, U., Pesch, E., and Phan-Huy, T. (2002). Constraint propagation and problem decomposition: a

preprocessing procedure for the job shop problem. *Annals of Operations Research*, 115: 125-145.
21. Eberhart, R.C. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, pp. 39-43.
22. Fisher, H. and Thompson, G.L. (1963). Probabilistic learning combinations of local job-shop scheduling rules. In: J.F. Muth and G.L. Thompson (Eds.), *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, pp. 225-251.
23. Garey, M., Johnson, D., and Sethy, R. (1976). The complexity of flow shop and job shop scheduling. *Mathematics of Operations Research*, 1: 117-129.
24. Groce, F.D., Tadei, R., and Volta, G. (1995). A genetic algorithm for the job shop problem. *Computers and Operations Research*, 22: 15-24.
25. Huang, W. and Yin, A. (2004). An improved shifting bottleneck procedure for the job shop scheduling problem. *Computers and Operations Research*, 31: 2093-2110.
26. Ikeda, K. and Kobayashi, S. (2000). GA based on the UV-Structure hypothesis and its application to JSP. In: *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN-VI)*, Springer-Verlag, Berlin, pp. 273-282.
27. Jain, A. and Meeran, S. (1999). Deterministic job-shop scheduling: past, present and future. *European Journal of Operations Research*, 113: 390-434.
28. Kennedy, J., Eberhart, R.C., and Shi, Y. (2001). *Swarm Intelligence*, San Mateo, Morgan Kaufmann, CA, USA.
29. Kolonko, M. (1999). Some new results on simulated annealing applied to the job shop scheduling problem. *European Journal of Operational Research*, 113: 123-136.
30. Lampinen, J. (2001). *A Bibliography of Differential Evolution Algorithm*, Technical Report, Laboratory of Information Processing, Department of Information Technology, Lappeenranta University of Technology.
31. Lawrence, S. (1984). *Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques (Supplement)*, Graduate School of Industrial Administration (GSIA), Carnegie-Mellon University, Pittsburgh, PA, USA.
32. Martin, P.D. (1996). *A Time-Oriented Approach to Computing Optimal Schedules for the Job-Shop Scheduling Problem*, Ph.D. Thesis, School of Operations Research and Industrial Engineering, Cornell University, NY, USA.
33. Masters, T. and Land, W. (1997). A new training algorithm for the general regression neural network, *Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation*, pp.1990-1994.
34. Mladenovic, N. and Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research,* 24: 1097-1100.
35. Murovec, B. and Šuhel, P. (2004). A repairing technique for the local search of the job-shop problem. *European Journal of Operational Research*, 153: 220-238.
36. Nowicki, E. and Smutnicki, C. (1996). A fast taboo search algorithm for the job shop problem. *Management Science*, 42: 797-813.
37. Onwubolu, G.C. and Clerc, M. (2004). Optimal operating path for automated drilling operations by a new heuristic approach using particle swarm optimisation. *International Journal of Production Research*, 42(3): 473-491.
38. Onwubolu, G.C. and Babu, B.V. (2004). *New Optimization Techniques in Engineering*, Springer Verlag.
39. Pezzella, F. and Merelli, E. (2000). A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research*, 120: 297-310.
40. Rae, A. and Parameswaran, S. (2001). Synthesising application-specific heterogenous multiprocessors using differential evolution. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E84-A(12): 3125-3131.
41. Rogalsky, T., Kocabiyik, S., and Derksen, R.W. (2000). Differential evolution in aerodynamic optimization. *Canadian Aeronautics and Space Journal*, 46(4): 183-190.
42. Ruzek, B. and Kvasnicka, M. (2001). Differential evolution algorithm in the earthquake hypocenter location. *Pure and Applied Geophysics*, 158(4), 667-693.
43. Salman, A., Ahmad, I., and Al-Madani, S. (2003). Particle swarm optimization for task assignment problem. *Microprocessors and Microsystems*, 26: 363-371.
44. Satake, T., Morikawa, K., Takahashi, K., and Nakamura, N. (1994). Neural network approach for minimizing the makespan of the general job-shop. *International Journal of Production Economics*, 33: 67-74.
45. Satake, T., Morikawa, K., Takahashi, K., and Nakamura, N. (1999). Simulated annealing approach for minimizing the makespan of the general job-shop. *International Journal of Production Economics*, 60-61: 515-522.
46. Steinhöfel, K., Albrecht, A., and Wong, C.K. (1999). Two simulated annealing-based heuristics for the job shop scheduling problem. *European Journal of Operational Research*, 118: 524-548.
47. Steinhöfel, K., Albrecht, A., and Wong, C.K. (2002). Fast parallel heuristics for the job shop scheduling problem. *Computers and Operations Research*, 29, 151-169.
48. Storn, R. (1999). Designing digital filters with differential evolution. In: D. Corne, M. Dorigo, and F. Glover (Eds.), *New Ideas in Optimization*, London: McGraw-Hill, UK, pp. 109-125.
49. 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, ICSI.
50. Storn, R. and Price, K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous space. *Journal of Global Optimization*, 11: 341-359.
51. Taillard, É. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64: 278-285.

52. Taillard, É. (1994). Parallel taboo search techniques for the job shop scheduling problem. *ORSA Journal on Computing*, 6(2), 108-117.
53. Tasgetiren, M.F. and Liang, Y.-C. (2003). A binary particle swarm optimization algorithm for lot sizing problem. *Journal of Economic and Social Research*, 5(2): 1-20.
54. Tasgetiren, M.F., Liang, Y.-C., Sevkli, M., and Gencyilmaz, G. (2004a). Particle swarm optimization algorithm for makespan and maximum lateness minimization in permutation flowshop sequencing problem. In *Proceedings of the 4th International Symposium on Intelligent Manufacturing Systems (IMS2004)*, Sakarya, Turkey, pp. 431-441.
55. Tasgetiren, M.F., Sevkli, M., Liang, Y.-C., and Gencyilmaz, G. (2004b). Differential evolution algorithm for permutation flowshop sequencing problem with makespan criterion. *Proceedings of the 4th International Symposium on Intelligent Manufacturing Systems* (*IMS2004*), Sakarya, Turkey, pp. 442-452.
56. Tasgetiren, M.F., Sevkli, M., Liang, Y.-C., and Gencyilmaz, G. (2004c). Particle swarm optimization algorithm for single machine total weighted tardiness problem. In *Proceedings of the 2004 Congress on Evolutionary Computation* (*CEC'04*), Portland, Oregon, pp. 1412-1419.
57. Tasgetiren, M.F., Sevkli, M., Liang, Y.-C., and Gencyilmaz, G. (2004d). Particle swarm optimization algorithm for permutation flowshop sequencing problem. In *Proceedings of the 4th International Workshop on Ant Colony Optimization and Swarm Intelligence* (*ANTS2004*), LNSC 3172, Brussels, Belgium, pp. 382-390.
58. Van den Bergh, F. and Engelbecht, A.P. (2000). Cooperative learning in neural networks using particle swarm optimizers. *South African Computer Journal*, 26: 84-90.
59. Van Laarhoven, P., Aarts, E., and Lenstra, J. (1992). Job shop scheduling by simulated annealing. *Operations Research*, 40: 113-125.
60. Wang, L. and Zheng, D.Z. (2001). An effective hybrid optimization strategy for job-shop scheduling problems. *Computers and Operations Research*, 28: 585-596.
61. Yamada, T. and Nakano, R. (1992). A genetic algorithm applicable to large-scale job shop problems. In *Proceedings of the Second International Conference on Parallel Problem Solving from Nature (PPSN-II)*, North-Holland, Amsterdam, pp 281-290.
62. Yamada, T. and Nakano, R. (1996). Job-shop scheduling by simulated annealing combined with deterministic local search. In I.H. Osman and J.P. Kelly (Eds.), *Meta-Heuristics: Theory and Applications*, Kluwer, 237-248.
63. Yeh, L.-W. (2003). *Optimal Procurement Policies for Multi-product Multi-supplier with Capacity Constraint and Price Discount*, Master thesis, Department of Industrial Engineering and Management, Yuan Ze University, Taiwan, R.O.C.
64. Yoshida, H., Kawata, K., Fukuyama, Y., and Nakanishi, Y. (2000). A particle swarm optimization for reactive power and voltage control considering voltage security assessment. *IEEE Transactions on Power Systems*, 15: 1232-1239.
65. Zhou, D.N., Cherkassky, V., Baldwin, T.R., and Olson, D.E. (1991). A Neural network approach to job-shop scheduling, *IEEE Transactions on Neural Networks*, 2(1): 175-179.