Particle Swarm Optimization Algorithm for Single Machine Total Weighted Tardiness Problem

M. Fatih Tasgetiren
Dept. of Management, Fatih University
34500 Buyukcekmece, Istanbul, Turkey
Email: ftasgetiren@fatih.edu.tr

Mehmet Sevkli
Dept. of Industrial Engineering, Fatih University
34500 Buyukcekmece, Istanbul, Turkey
Email: msevkli@fatih.edu.tr

Yun-Chia Liang
Dept. of Industrial Engineering and Management
Yuan Ze University
No 135 Yuan-Tung Road, Chung-Li,
Taoyuan County, Taiwan 320, R.O.C.
Email: ycliang@saturn.yzu.edu.tw

Gunes Gencyilmaz
Dept. of Management,
Istanbul Kultur University
E5 Karayolu Uzeri, Sirinevler, Istanbul, Turkey
Email: g.gencyilmaz@iku.edu.tr

***Abstract*: In this paper we present a particle swarm optimization algorithm to solve the single machine total weighted tardiness problem. A heuristic rule, the *Smallest Position Value (SPV)* rule, is developed to enable the continuous particle swarm optimization algorithm to be applied to all classes of sequencing problems, which are NP-hard in the literature. A simple but very efficient local search method is embedded in the particle swarm optimization algorithm. The computational results show that the particle swarm algorithm is able to find the optimal and best-known solutions on all instances of widely used benchmarks from the OR libary.**

I. INTRODUCTION

Particle Swarm Optimization (PSO) is one of the latest population-based optimization methods, which 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. In a PSO algorithm, each member is called "particle", and each particle flies around in the multi-dimensional search space with a velocity, which is constantly updated by the particle's own experience and the experience of the particle's neighbors or the experience of the whole swarm. Two variants of the PSO algorithm are developed, namely PSO with a local neighborhood, and PSO with a global neighborhood. According to the global neighborhood, each particle moves towards its best previous position and towards the best particle in the whole swarm, called *gbest* model. On the other hand, according to the local variant so called *lbest*, each particle moves towards its best previous position and towards the best particle in its restricted neighborhood [1].

Since PSO was first introduced by Kennedy and Eberhart [2, 3], it has been successfully applied to optimize various continuous nonlinear functions. Although the applications of PSO on combinatorial optimization problems are still limited, PSO has certain advantages such as easy to implement and computationally efficient. Therefore, this paper is the first to employ PSO on solving single machine total weighted tardiness (SMTWT) problem which is a typical combinatorial optimization problem.

McNaughton [4] first presented a scheduling problem that the objective is to minimize total penalty cost. He proved that an optimal solution exists in which no job is split, so that only permutation schedules of the n jobs need to be considered. Therefore, the SMTWT problem can be stated as follows. Each of n jobs ($j = 1,...,n$) is to be processed without preemption on a single machine that can handle no more than one job at a time. The processing and set-up requirements of any job are independent of its position in the sequence. The release time of all jobs is zero. Thus, job j ($j = 1,...,n$) becomes available for processing at time zero, requires an uninterrupted positive processing time p_j , which includes set-up and knock-down times on the machine, has a positive weight w_j , and has a due date d_j by which job j should ideally be finished. For a given processing

0-7803-8515-2/04/$20.00 ©2004 IEEE

order of all jobs, the completion time C_j and the tardiness $T_j = \max\{0, C_j - d_j\}$ of job j can be computed. Then, the SMTWT problem is to develop a sequence of jobs that minimizes the sum of the weighted tardiness $\sum_{j=1}^{n} w_j T_j$ over all jobs.

The following paper is organized as follows. Section II gives a brief review of SMTWT problem and PSO literatures. In section III, methodology of the PSO algorithm is discussed, and computational results of test problems are shown in section IV. Finally, section V summarizes the concluding remarks.

II. LITERATURE REVIEW

A. Single Machine Total Weighted Tardiness (SMTWT) Problem

The solving techniques of SMTWT can be broadly classified into three main categories: exact solutions methods, dispatching rules, and heuristic approaches. Exact solutions methods consist of dynamic programming [5-9] and branch-and-bound algorithms [9-12]. Both use dominance rules to restrict the search for the optimal solution and cause computational practicality. Rinnooy Kan et al. [11] extend Emmons' dominance rules [13] to the SMTWT problem. None of the DP algorithms can consistently solve problems with more than 30 jobs within an acceptable computation time. Branch-and-bound approaches, in general, also require very large computation times when the number of jobs is more than 50. Recent state-of-art survey [14] provides an excellent review of static scheduling to minimize weighted and unweighted tardiness.

A variety of dispatching rules have been proposed to solve problems quickly. Potts and van Wassenhove [15] compare weighted shortest processing time (WSPT), earliest due date (EDD), modified cost over time (MCOVERT), and apparent urgency (AU), and find that AU performs best and WSPT is greatly inferior. Alidaee and Ramakrishnan [16] provide computational tests of the COVERT-AU class of dispatching rules for problems of up to 200 jobs. Dispatch rules offer a quick sequencing method, but have poor worst-case performance.

Since there is no dominant method for all problem environments, researchers have paid more attention to heuristic methods, such as simulated annealing (SA) [17, 18], tabu search (TS) [18], GA [18], and ant colony optimization (ACO) [19-21], which offer a compromise between computational expense and solution quality. Crauwels et al. [18] compare several heuristic methods including descent, threshold accepting, TS, SA, and GA. TS is superior to other methods, although the binary encoded GA generates solutions with good quality as well. Congram et al. [22] propose an iterated dynasearch algorithm for the SMTWT problem. The authors develop an approach which allows a series of moves to be performed at each iteration while traditional local search makes a single move only. Problem size up to 100 jobs is tested and this iterated algorithm outperforms the best algorithm, TS, in [18]. ACO by far is the best heuristic methods on solving the SMTWT problem. Den Besten et al. [19] and Liang [21] both apply an ant colony system (ACS) to a set of test problems of size up to 100 jobs from the literature with excellent results. Merkle et al. [20] use the pheromone summation evaluation for the transition probability and a new problem specific heuristic. The authors test their algorithm in both weighted and unweighted problems, and the results are worse than previous two ACO algorithms.

B. Particle Swarm Optimization (PSO)

PSO is one of the latest evolutionary optimization techniques, and it is based on the metaphor of social interaction and communication such as bird flocking and fish schooling. Since it is population-based and evolutionary in nature, the members in a PSO algorithm tend to follow the leader of the group, i.e., the one with the best performance. The details of PSO algorithm will be described in the following section. Although PSO is still new in evolutionary computation field, it has been applied to a wide a range of applications such as power and voltage control [23], optimal power flow design [27], and task assignment problem [28]. More literature can be found in references [1] and [25].

III. PSO ALGORITHM FOR SMTWT PROBLEM

Evaluation of each particle in the swarm requires the determination of the permutation of jobs for the PFSP since the total weighted tardiness value is a result of the sequence. In this paper, we propose a heuristic rule called *Smallest Position Value (SPV)* to enable the continuous PSO algorithm to be applied to all class of sequencing problems, which are NP-hard in the literature. By using the *SPV* rule,

the permutation can be determined through the position values of the particle so that the fitness value of the particle can then be computed with that permutation. We follow the *gbest* model of Kennedy et al. [1] with the inclusion of SPV rule in the algorithm. Pseudo code of the PSO algorithm for the PFSP is given in Figure 1.

Initialize parameters
Initialize population
Find sequence
Evaluate
Do { Find the personal best
Find the global best
Update velocity
Update position
Find sequence
Evaluate
Apply local search (optional)
} While (Termination)

Figure 1. PSO Algorithm with Local Search for SMTWT Problem.

The basic elements of PSO algorithm is summarized as follows:

Particle: X_i^k denotes the i^{th} particle in the swarm at iteration k and is represented by n number of dimensions as $X_i^k = \left[x_{i1}^k,..,x_{in}^k\right]$, where x_{ij}^k is the position value of the i^{th} particle with respect to the j^{th} dimension ($j = 1,..,n$).

Population: pop^k is the set of m particles in the swarm at iteration k, i.e., $pop^k = \left[X_1^k,..,X_m^k\right]$

Sequence: We introduce a new variable S_i^k, which is a sequence of jobs implied by the particle X_i^k. It can be described as $S_i^k = \left[s_{i1}^k,..,s_{in}^k\right]$, where s_{ij}^k is the assignment of job j of the particle i in the processing order at iteration k with respect to the j^{th} dimension

Particle velocity: V_i^k is the velocity of particle i at iteration k. It can be defined as $V_i^k = \left[v_{i1}^k,..,v_{in}^k\right]$, where v_{ij}^k is the velocity of particle i at iteration k with respect to the j^{th} dimension.

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

Personal best: PB_i^k represents the best position of the particle i with the best fitness until iteration k. So, the best position associated with the best fitness value of the particle i obtained so far is called the *personal best*. For each particle in the swarm, PB_i^k can be determined and updated at each iteration k. In a minimization problem with the objective function $f\left(S_i^k\right)$ where S_i^k is the corresponding sequence of particle X_i^k, the personal best PB_i^k of the i^{th} particle is obtained such that $f\left(S_i^k\right) \le f\left(S_i^{k-1}\right)$ where S_i^k is the corresponding sequence of personal best PB_i^k and S_i^{k-1} is the corresponding sequence of personal best PB_i^{k-1}. To simplify, we denote the fitness function of the personal best as $f_i^{pb} = f\left(S_i^k\right)$. For each particle, the personal best is defined as $PB_i^k = \left[pb_{i1}^k,..,pb_{in}^k\right]$ where pb_{ij}^k is the position value of the i^{th} personal best with respect to the j^{th} dimension.

Global best: GB^k denotes the best position of the globally best particle achieved so far in the whole swarm. For this reason, the global best can be obtained such that $f\left(S^k\right) \le f\left(S_i^k\right)$ for $i = 1,2,..,m$ where S^k is the corresponding sequence of global best GB^k and S_i^k is the corresponding sequence of particle best PB_i^k .To simplify, we denote the fitness function of the global best as $f^{gb} = f\left(S^k\right)$. The global best is then defined as $GB^k = \left[gb_1^k,..,gb_n^k\right]$ where gb_j^k is the position value of the global best with respect to the j^{th} dimension.

Termination criterion: It is a condition that the search process will be terminated. It might be a maximum number of iteration or maximum CPU time to terminate the search.

A. Solution Representation

One of the most important issue when designing the PSO algorithm lies on its solution representation. In order to construct a direct relationship between the problem domain and the PSO particles for the SMTWT problem, we present n number of dimensions for n number of jobs ($j = 1,..,n$). In other words, each dimension represents a typical job. In addition, the particle $X_i^k = \left[x_{i1}^k,..,x_{in}^k\right]$ corresponds to the position values for n number of jobs in the SMTWT problem. Each particle has a continuous set of position values. The particle itself

does not present a sequence. Instead we use the SPV rule to determine the processing sequence implied by the position values x_{ij}^k of particle X_i^k . Table 1 illustrates the solution representation of particle X_i^k for the PSO algorithm with its velocity and corresponding sequence. According to the proposed SPV rule, the smallest position value is $x_{i5}^k = -1.20$, so the dimension *j*=5 is assigned to be the first job $s_{i1}^k = 5$ in the processing sequence S_i^k ; the second smallest position value is $x_{i2}^k = -0.99$, so the dimension *j*=2 is assigned to be the second job $s_{i2}^k = 2$ in the processing sequence S_i^k , and so on. In other words, dimensions are sorted according to the SPV rule, i.e., to x_{ij}^k values to construct the sequence S_i^k . This representation is unique in terms of finding new solutions since positions of each particle are updated at each iteration *k* in the PSO algorithm, thus resulting in different sequences at each iteration *k*.

Table 1. Solution Representation of a Particle.

j	1	**2**	3	4	**5**	6
x_{ij}^k	1.80	**-0.99**	3.01	-0.72	**-1.20**	2.15
v_{ij}^k	3.89	2.94	3.08	-0.87	-0.20	3.16
s_{ij}^k	**5**	**2**	4	1	6	3

B. Initial Population

A population of particles is constructed randomly for the PSO algorithm of the SMTWT problem. The continuous values of positions are established randomly. The following formula is used to construct the initial continuous position values of the particle: $x_{ij}^0 = x_{\min} + (x_{\max} - x_{\min}) * U(0,1)$ where $x_{\min} = 0.0, x_{\max} = 4.0,$ and $U(0,1)$ is a uniform random number between 0 and 1. Initial continuous velocities are generated by similar formula as follows:

$v_{ij}^0 = v_{\min} + (v_{\max} - v_{\min}) * U(0,1)$ where $v_{\min} = -4.0, v_{\max} = 4.0,$ and $U(0,1)$ is a uniform random number between 0 and 1. Continuous velocity values are restricted to some range, namely $v_{ij}^k = [v_{\min}, v_{\max}] = [-4.0, 4.0]$ where $v_{\min} = -v_{\max}$.

As the formulation of the SMTWT problem suggests that the objective is to minimize the total weighted tardiness, the fitness function value is the total weighted tardiness for the particle *i* of the population or swarm. That is,

$$f_i^k(S_i^k) = \sum_{j=1}^{n} w_{ij} T_{ij}$$

Where S_i^k is the corresponding sequence of particle X_i^k . For simplicity, $f_i^k(S_i^k)$ will be denoted as f_i^k . The complete computational flow of the PSO algorithm for the SMTWT problem can be summarized as follows:

Step 1: Initialization

- Set *k*=0, *m*=twice the number of dimensions.
- Generate *m* particles randomly as explained before, $\{X_i^0, i = 1,..,m\}$ where $X_i^0 = [x_{i1}^0,..,x_{in}^0]$.
- Generate initial velocities of particles randomly $\{V_i^0, i = 1,..,m\}$ where $V_i^0 = [v_{i1}^0,..,v_{in}^0]$
- Apply the SPV rule to find the sequence $S_i^0 = [s_{i1}^0,..,s_{in}^0]$ of particle X_i^0 for $i = 1,..,m$.
- Evaluate each particle *i* in the swarm using the objective function f_i^0 for $i = 1,..,m$.
- For each particle *i* in the swarm, set $PB_i^0 = X_i^0$, where $PB_i^0 = [pb_{i1}^0 = x_{i1}^0,.., pb_{in}^0 = x_{in}^0]$ along with its best fitness value, $f_i^{pb} = f_i^0$ for $i = 1,..,m$.
- Find the best fitness value $f_l^0 = \min\{f_i^0\}$ for $i = 1,..,m$ with its corresponding position X_l^0 .
- Set global best to $GB^0 = X_l^0$ where $GB^0 = [gb_1 = x_{l,1},.., gb_n = x_{l,n}]$ with its fitness value $f^{gb} = f_l^0$

Step 2: Update iteration counter

- $k = k + 1$

Step3: Update inertia weight

- $w^k = w^{k-1} * \alpha$ where α is decrement factor.

Step 4: Update velocity

$$v_{ij}^k = w^{k-1} v_{ij}^{k-1} + c_1 r_1 (pb_{ij}^{k-1} - x_{ij}^{k-1}) + c_2 r_2 (gb_j^{k-1} - x_{ij}^{k-1})$$

Step 5: Update position

- $x_{ij}^k = x_{ij}^{k-1} + v_{ij}^k$

Step 6: Find Sequence

- Apply the SPV rule to find the sequence $S_i^k = \left[s_{i1}^k,..,s_{in}^k\right]$ for $i = 1,..,m$.

Step 7: Update personal best

- Each particle is evaluated by using its sequence to see if personal best will improve. That is, if $f_i^k < f_i^{pb}$ for $i = 1,..,m$, then personal best is updated as $PB_i^k = X_i^k$ and $f_i^{pb} = f_i^k$ for $i = 1,..,m$.

Step 8: Update global best

- Find the minimum value of personal best. $f_l^k = min\left\{f_i^{pb}\right\} for \quad i = 1,..,m, l \in \left\{i; i = 1,..,m\right\}$
- If $f_l^k < f^{gb}$, then the global best is updated as $GB^k = X_l^k$ and $f^{gb} = f_l^k$

Step 9: Stopping criterion

- If the number of iteration exceeds the maximum number of iteration, or maximum CPU time, then stop, otherwise go to step 2.

C. Local Search for SMTWT Problem

In the proposed PSO algorithm, local search is applied to the sequence directly. However, it violates the SPV first rule and needs a repair algorithm. This approach is illustrated in Table 2 and 3, where job $s_{i2}^k = 2$ and job $s_{i4}^k = 1$ are interchanged.

Table 2. Local Search Applied to Sequence Before Repairing

j	1	2	3	4	5	6
x_{ij}^k	1.80	-0.99	3.01	-0.72	-1.20	2.15
Job, s_{ij}^k	5	**2**	4	**1**	6	3
x_{ij}^k	1.80	-0.99	3.01	-0.72	-1.20	2.15
Job, s_{ij}^k	5	**1**	4	**2**	6	3

As seen in Table 2, applying a local search to the sequence violates the SPV rule because the sequence itself is a result of the particle's position values. Once a local search is completed, particle should be repaired so that the SPV is not violated. This is achieved by changing the position values according to the SPV rule as shown in Table 3.

In other words, interchange the position values of the exchanged jobs in terms of their dimensions. Since jobs $s_{i2}^k = 2$ and $s_{i4}^k = 1$ are interchanged, their associated position values $x_{i2}^k = -0.99$ and $x_{i1}^k = 1.80$ are interchanged for dimensions j=2 and j=1 to keep the particle consistent with the SPV rule. The advantage of this method is due to the fact that the repair algorithm is only needed after evaluating all the neighbors in a sequence. Second approach is employed in this work.

Table 3. Local Search Applied to Sequence after Repairing

j	**1**	**2**	3	4	5	6
x_{ij}^k	**1.80**	**-0.99**	3.01	-0.72	-1.20	2.15
Job, s_{ij}^k	5	**2**	4	**1**	6	3
x_{ij}^k	**-0.99**	**1.80**	3.01	-0.72	-1.20	2.15
Job, s_{ij}^k	5	**1**	4	**2**	6	3

The performance of the local search algorithm depends on the choice of the neighborhood structure. For the SMWTP, following two neighborhood structures are employed as in [19, 24]:

- Interchange two jobs between u^{th} and v^{th} dimensions, $u \neq v$ *(Interchange)*
- Remove the job at the u^{th} dimension and insert it in the v^{th} dimension $u \neq v$ *(Insert)*

Local search in this work is based on the insert+interchange variant of the variable neighborhood search (VNS) method presented in [26]. Pseudo code of the local search is given in Figure 2 where u and v are the integer random numbers between 1 and n. π=*insert*(π_0, u, v) means removing the job from u^{th} dimension in the sequence π_0 and inserting it in the v^{th} dimension in the sequence π_0, thus resulting in a sequence π. The probability of the local search, p_{ls} , is taken as 10 percent. In other words, the local search is applied to the sequence S^k of global best solution GB^k at each iteration p_{ls} times population size.

```
outloop=0;
do{ π0= S^k  , sequence of global best GB^k ;
   u=rnd(1,n); v=rnd(1,n);
  π=insert(π0, u, v)
   inloop=0;
   do{ kcount=0; max_method=2;
      do{ u=rnd(1,n); v=rnd(1,n);
         if (kcount=0) then π1=insert(π, u, v)
         if (kcount=1) then π1=interchange(π, u, v)
       if(f(π1)<f(π))then{kcount=0; π= π1;}
       else { kcount++;}
```

```
    while (kcount<max_method)
  inloop++;
  while (inloop<n*(n-1)
  outloop++;
  if (f(π)<=f(S^k)) then {S^k =π;
                          repair(GB^k);}
while (outloop< p_ls * popsize);
```

Figure 2. Pseudo Code of Local Search for SMTWT Problem.

IV. EXPERIMENTAL RESULTS

The proposed PSO algorithm presented for the SMTWT problem is coded in C and run on an Intel Pentium IV 2.6GHz PC with 256MB memory. The performance of the pure PSO, denoted as PSO_{spv}, is first presented and then PSO with the local search, denoted as PSO_{vns}, is compared with the best reported results in the literature [19, 24]. We used the following parameters for the PSO. The size of the population is taken as the twice the number of dimensions. Social and cognitive parameters, and uniform random numbers are taken as $c_1 = c_2 = 2$ and $r_1 = r_2 = 0.5$ respectively. Initial inertia weight is set to $w^0 = 0.9$ and never decreased below 0.40. Finally, the decrement factor α is taken as 0.975.

We evaluated the performance of the proposed PSO algorithm using the benchmark set of randomly generated instances, available on the ORLIB (http://mscmga.ms.ic.ac.uk/jeb/orlib/wtinfo.html). There exist three sets of problems with 40, 50 and 100 jobs, each containing 125 instances. The optimal solutions for the 40 and 50 job instances are known. However, the 100 job instances are not solved to optimality so far and best-known solutions are available.

The best reported results so far in the literature are the *Ant Colony Optimization* (ACO_{vnd}) [19] and *Iterative Local Search* (ILS_{vnd}) [24] which they found the best-known solutions of 125 instances for the difficult 100 job problems. Both ACO_{vnd} and ILS_{vnd} has employed the Variable Neighborhood Descent (VND) method as a local search in their algorithm. First, we give the results for the PSO_{spv} algorithm to see the performance of the SPV rule we proposed. Then we compare the PSO_{vns} to ACO_{vnd} and ILS_{vnd}. In both approach, 25 replications are conducted for each instance with a large computation time so that each instance could be solved to the optimal or best known-solution, which they conjecture to be optimal, in each single replication. Both algorithms found the best-known solutions in reasonable CPU times.

For the evaluation of the performance of the SPV rule, the maximum CPU times are set to 5 seconds for 40 job problems, 10 seconds for 50 job problems and 100 seconds for 100 job problems to see the impact of the SPV rule on the solution quality, and 25 replications are made for each instance. The results are given in Table 4 where Δ_{avg} denotes the average relative percent deviation from the optimal or best-known solutions, $\Delta_{\max}$ denotes the maximum relative percent deviation from the optimal or best-known solutions, n_{opt} denotes the number of optimal or best-known solutions found, and t_{avg} ., $t_{\min}$. and $t_{\max}$ denotes the average, minimum and maximum CPU times respectively, finally σ_t denotes the standard deviation of the CPU times. Δ_{avg} is computed as follows:

$$\Delta_{avg} = \sum_{i=1}^{R} \left(\frac{heuristic_i - bestknown_i}{bestknown_i} \right) / R$$

where R is the total number of replications, i.e., R is equal to the number of replications for each instance times the total number of instances. $\Delta_{\max}$ is the maximum relative percent deviation of R runs.

Table 4. Impact of SPV Rule

# of Job	t_{avg}	$\Delta_{\max}$	n_{opt}	t_{avg}	σ_t	$t_{\min}$	$t_{\max}$
40	0.02	5.68	120	2.84	2.28	0.00	5.0
50	0.02	1.28	110	6.87	4.31	0.01	10.0
100	0.04	12.73	51	77.97	39.48	0.02	100

According to the results, PSO_{spv} is able to find 120 optimal solutions out of 125 instances for 40 job problems, 110 optimal solutions out of 125 instances for 50 job problems and 51 best-known solutions out of 125 instances for the difficult 100 job problems. In terms of the average percentage deviation Δ_{avg}, the PSO_{spv} is able to find the solutions with a 2% deviation from the optimal solutions for the 40 and the 50 job problems, and 4% deviation from the best known solutions for the 100-job problems.

In the second case, we used the iterated *insert+interchange* VNS as a local search in the PSO algorithm. The iterated *insert+interchange* VNS is applied to the sequence of *global best* solution at each iteration in a way that the sequence of global best solution is first modified with insert operator and then VNS local search is applied, and this process is repeated 10 percent times population size. Again, the maximum CPU times are set to 5 seconds for 40 job problems, 10 seconds for 50 job problems and 100 seconds for 100 job problems and 25 replications are made for each instance to collect the statistics.

In Table 5, the statistics regarding the performance of PSO_{vns} ACO_{vnd} and ILS_{vnd} algorithms are presented for comparison purposes. In terms of n_{opt}, three algorithms are able to find the optimal or best-known solution for all the problem instances. In terms of the overall average of 40, 50 and 100 jobs for t_{avg}, PSO_{vns} is computationally less expensive than both approaches, for σ_t, PSO_{vns} is more robust than ILS_{vnd} but is inferior to ACO_{vnd}, for $t_{\min}$, PSO_{vns} is not able to find the solutions quickly especially for 100 job problems, for $t_{\max}$, ACO_{vnd} is computationally less expensive to find the best known solutions for the hardest instances whereas both PSO_{vns} and ILS_{vnd} responded to the hardest instances almost equally in terms of computational time. Since different machines are used in three different approaches, the comparisons based on the CPU times might not seem to be meaningful. However, three approaches are able to find the optimal and best known solutions for all the problem instances in reasonable CPU times. In summary, one can conclude that the PSO_{vns} is as good as the ant colony and iterated local search algorithms.

An analysis of Tables 4 and 5 shows that employing the VNS in the PSO algorithm improved the performance of the solution quality especially for the 100 job difficult instances significantly.

V. CONCLUSIONS

To the best of our knowledge, this is the first reported application of the particle swarm optimization algorithm to the single machine total weighted tardiness problem in the literature. We present the SPV rule to make the continuous PSO algorithms to be able to be applied to all class of sequencing and scheduling problems. The main disadvantage of the PSO algorithms is the quick convergence. Making the PSO converge slowly can improve the performance of the SPV rule employed to solve the sequencing problems. PSO research is still at the pioneering stage and lots of researches are required to investigate the impact of employing some operators such as fitness scaling, mutation, immigration, constriction factor, craziness operator and so on the solution quality.

Table 5. Comparison of PSO with ACO and ILS

	Job	n_{opt}	t_{avg}	σ_t	$t_{\min}$	$t_{\max}$
ACO_{vnd}	40	125	0.088	0.021	0.0040	1.72
	50	125	0.320	0.869	0.0060	10.74
	100	125	6.990	7.019	0.0180	86.26
Overall Average		125	2.466	2.636	0.0093	32.91
ILS_{vnd}	40	125	0.040	0.130	0.0110	1.23
	50	125	0.200	0.860	0.0017	8.71
	100	125	5.750	14.500	0.0076	105.56
Overall Average		125	1.997	5.163	0.0068	38.50
PSO_{vns}	40	125	0.088	0.162	0.0000	5.00
	50	125	0.264	0.680	0.0930	10.00
	100	125	4.237	10.018	1.202	100.0
Overall Average		125	1.530	3.620	0.432	38.33

In summary, the results presented in this work is very encouraging and promising for the application of the PSO with SPV rule to the single machine total weighted tardiness problem, hence to the other scheduling problems. The main contribution of this paper is to provide a way to apply the continuous PSO algorithms to the discrete combinatorial optimization problems

For the future work, we will extend this work to the larger instances to evaluate the impact of the SPV rule together with the VNS local search. No speed-up techniques and problem specific information are used in the PSO presented in this paper. Inclusion of speed up techniques described in [22] and problem specific information in the initial population may help the PSO to tackle the larger instances.

REFERENCES

[1] J. Kennedy, R. C. Eberhart and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann, San Mateo, CA, 2001.

[2] R. C. Eberhard and J. Kennedy, "A new optimizer using particle swarm theory," *Proc. of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, pp. 39-43, 1995.

[3] J. Kennedy and R. C. Eberhard, "Particle swarm optimization," *Proc. of IEEE Int'l Conf. on Neural Networks*, Piscataway, NJ, USA, pp. 1942-1948, 1995.

[4] R. McNaughton, "Scheduling with deadlines and loss functions," *Management Science*, vol. 6, no. 1, pp. 1-12, 1959.

[5] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 10, no. 1, pp. 196-210, 1962.

[6] E. L. Lawler, "On scheduling problems with deferral costs," *Management Science*, vol. 11, no. 2, pp. 280-288, 1964.

[7] K. R. Baker and L. E. Schrage, "Finding an optimal sequence by dynamic programming: an extension to precedence-related tasks," *Operations Research*, vol. 26, no. 1, pp. 111-120, 1978.

[8] L. E. Schrage and K. R. Baker, "Dynamic programming solution of sequencing problems with precedence constraints," *Operations Research*, vol. 26, no. 3, pp. 444-449, 1978.

[9] T. S. Abdul-Razaq, C. N. Potts, and L. N. Van Wassenhove, "A survey of algorithms for the single machine total weighted tardiness scheduling problems," *Discrete Applied Mathematics*, vol. 26, pp. 235-253, 1990.

[10] J. Shwimer, "On the n-job, one-machine, sequence-independent scheduling problem with tardiness penalties: a branch-bound solution," *Management Science*, vol. 18, no. 6, pp. B301-B313, 1972.

[11] A. H. G. Rinnooy Kan, B. J. Lageweg, and J. K. Lenstra, "Minimizing total costs in one-machine scheduling," *Operations Research*, vol. 23, no. 5, pp. 908-927, 1975.

[12] C. N. Potts and L. N. Van Wassenhove, "A branch and bound algorithm for the total weighted tardiness problem," *Operations Research*, vol. 33, no. 2, pp. 363-377, 1985.

[13] H. Emmons, "One-machine sequencing to minimize certain functions of job tardiness," *Operations Research*, vol. 17, no. 4, pp. 701-715, 1969.

[14] S. Tapan, J. M. Sulek, and P. Dileepan, "Static scheduling research to minimize weighted and unweighted tardiness," *International Journal of Production Economics*, vol. 83, pp. 1-12, 2003.

[15] C. N. Potts and L. N. Van Wassenhove, "Single machine tardiness squencing heuristics," *IIE Transactions*, vol. 23, no. 4, pp. 346-354, 1991.

[16] B. Alidaee and K. R. Ramakrishnan, "A computational experiment of COVERT-AU class of rules for single machine tardiness scheduling problem," *Computers and Industrial Engineering*, vol. 30, no. 2, pp. 201-209, 1996.

[17] H. Matsuo, C. J. Suh, and R. S. Sullivan, "A controlled search simulated annealing method for the single machine weighted tardiness problem," *Annals of Operations Research*, vol. 21, pp. 85-108, 1989.

[18] H. A. Crauwels, C. N. Potts, and L. N. Van Wassenhove, "Local search heuristics for the single machine total weighted tardiness scheduling problem," *INFORMS Journal on Computing*, vol. 10, no. 3, pp. 341-350, 1998.

[19] M. den Besten, T. Stutzle and M. Dorigo, "Ant colony optimization for the total weighted tardiness problem," *Proc. of PPSN-VI: Sixth International Conf. on Parallel Problem Solving from Nature*, LNCS 1917, pp. 611-620, 2000.

[20] D. Merkle and M. Middendorf, "An ant algorithm with a new pheromone evaluation rule for total tardiness problems," *Proc. of EvoWorkshops 2000: Real-World Applications of Evolutionary Computing*, LNCS 1803, pp. 287-296, 2000.

[21] Y.-C. Liang, "Ant colony optimization approach to combinatorial problems," Ph.D. dissertation, Department of Industrial and Systems Engineering, Auburn University, 2001.

[22] R. K. Congram, C. N. Potts, and S. L. Van de Velde, "An iterated dynasearch algorithm for the single machine total weighted tardiness scheduling problem," *Technical Report*, Faculty of Mathematical Studies, University of Southampton, 1998.

[23] H. Yoshida, K. Kawata, Y. Fukuyama, and Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," *IEEE Transactions on Power Systems*, vol. 15, pp.1232-1239, 2000.

[24] M. den Besten, T. Stutzle, and M. Dorigo, "Design of iterated local search algorithms", In *Proc. of Evo Workshops 2001,* LNCS 2037, Springer, Berlin, pp. 441-451, 2001.

[25] *Proc. of 2003 IEEE Swarm Intelligence Symposium*, Indianapolis, IN, April, 2003.

[26] N. Mladenovic and P. Hansen, "Variable Neighborhood Search", *Computers and Operations Research*, vol. 24, pp. 1097-1100, 1997.

[27] M. A. Abido, "Optimal power flow using particle swarm optimization*", Electrical Power and Energy Systems*, vol. 24, pp. 563-571, 2002.

[28] A. Salman, I. Ahmad, S. Al-Madani, ":Particle swarm optimization for task assignment problem", *Microprocessors and Microsystems*, vol. 26, pp. 363-371, 2003.