

A Particle Swarm Optimization Algorithm for Permutation Flow Shop Sequencing Problem with the Number of Tardy Jobs Criterion

Hatice Ucar, Industrial & Manufacturing Engineering Department, Wayne State University, Detroit, MI, 48202, USA
M. Fatih Tasgetiren, Department of Management, Fatih University, Buyukcekmece, Istanbul, 34500, Turkey

In this study, we proposed a discrete particle swarm optimization algorithm to determine a sequence of n jobs to be processed through m machines that minimizes the number of tardy jobs. We also developed a traditional genetic algorithm and compared the performances of both algorithms for finding the optimal processing sequences. The algorithms were implemented using the due date configurations of Demirkol et al. (1998)'s data sets. It is concluded that the particle swarm optimization algorithm gives promising solutions by means of the proposed SPV (Smallest Position Value) heuristic rule.

Keyword: Scheduling, Number of Tardy Jobs, Permutation Flow Shop, Particle Swarm Optimization, Smallest Position Value, Genetic Algorithm

1. Introduction

Scheduling is of great importance in both manufacturing and service industries. It significantly improves the utilization of resources and profitability of production lines. It has many applications ranging from distribution networks to machine environment. In this study, we deal with machine scheduling in permutation flow shops.

Flow-shop scheduling is one of the most well-known problems in the area of scheduling. Various dispatching rules, exact, and heuristic methods have been proposed since the pioneering work of Johnson (1954) for the job sequencing problems on single or two machine environments, whereas the examples of flow shop scheduling are too few; the available ones are mostly concerned with the makespan and/or maximum lateness minimization.

Hariri and Potts (1989) developed the first exact algorithm for the number of tardy jobs problem for permutation flow shops, $Fm | \sum U_i$; however, it is not very practical for large sized flow shops. In this study, we are going to solve this particular problem using a traditional genetic algorithm (GA) and a particle swarm optimization algorithm (PSO). Even though GA is mostly used in scheduling problems, we will be the first to apply PSO to the $Fm | \sum U_i$ problem.

The paper is organized as follows: Section 2 gives a brief description of number of tardy jobs problem in a permutation flow shop. In section 3, literature review related to the number of tardy job problem is given. The genetic and particle swarm optimization algorithms and their methodologies are introduced in section 4 and 5, respectively. The experimental design

of the studies and their computational results are shown in sections 6 and 7. Finally, we draw conclusions from results in section 8.

2. Problem Definition

In a flow shop manufacturing system, different machines are set up in series, and each task is performed on all machines. There are n number of jobs available to be processed on m number of machines. If a flow shop is a permutation type, a sequence for n jobs is determined, and the jobs undergo through operation on all machines without changing their sequence.

The assumptions for the permutation flow shop problem are the following: 1) Each task is to be processed once on each machine from machine 1 to m . 2) Each task is done on at most one machine at a time. 3) Each machine can process only one task at a time. 4) Preemption is not allowed. 5) Set-up times are negligible. 6) All tasks release at work center at time zero. 7) The operating sequence of tasks is the same on every machine.

Minimizing the number of tardy jobs is important for the manufacturer to avoid the loss of customer good will; therefore, it is of great interest in industry. Our problem in this study is the minimization of number of tardy jobs in permutation flow shops.

To be able to define a job as tardy or early, we have to compute the completion time and the tardiness of each job. Given the processing times p_{jk} for job j and machine k , and a job permutation $\pi = \{\pi_1, \pi_2, \dots, \pi_n\}$ where n jobs ($j = 1, 2, \dots, n$) will be sequenced through m machines ($k = 1, 2, \dots, m$) using the same permutation, then the problem is to find the best permutation of jobs to be valid for each machine. If we label $C(\pi_j, m)$ as the completion time of job π_j on machine m , given the job permutation $\pi = \{\pi_1, \pi_2, \dots, \pi_n\}$, the calculation of completion time for n -job m -machine problem is given as follows:

$$\begin{aligned} C(\pi_1, 1) &= p_{\pi_1, 1} \\ C(\pi_j, 1) &= C(\pi_{j-1}, 1) + p_{\pi_j, 1} & j = 2, \dots, n \\ C(\pi_1, k) &= C(\pi_1, k-1) + p_{\pi_1, k} & k = 2, \dots, m \\ C(\pi_j, k) &= \max\{C(\pi_{j-1}, k), C(\pi_j, k-1) + p_{\pi_j, k}\} & j = 2, \dots, n; k = 2, \dots, m. \end{aligned}$$

Let $T(\pi_j)$ denote the tardiness of job π_j and can be defined as

$$T(\pi_j) = C(\pi_j, m) - d(\pi_j)$$

where $d(\pi_j)$ is the due date of job π_j , and if we label $ck(\pi)$ as the total number of tardy jobs counter for permutation π , π^* as the sequence giving the minimum number of tardy jobs, and N as the number of iterations; then,

$$U_j = \begin{cases} 0, & \text{if } T(\pi_j) \leq 0 \\ 1, & \text{if } T(\pi_j) > 0 \end{cases}$$

$$c_0(\pi) = 0,$$

```

for(k = 1, k <= N, k++)
for(j = 1, j <= n, j++)
if(Uj = 1 => ck(π) = ck(π)+1;
for(k = 1, k <= N, k++)
π* = arg min(ck(π))

```

3. Literature Review

Single machine scheduling constitutes the widest part of the literature. There are also examples of two-machine scheduling; but examples of m-machine environment in the literature are too few.

Moore (1968) found that minimizing the number of tardy jobs on a single machine, $F1 | \sum U_i$ is an easy problem which requires $O(n \log n)$ time to be solved, whereas Karp (1972) demonstrates that minimizing the weighted number of tardy jobs on a single machine, $F1 | \sum w_i U_i$ is NP-hard. Single machine problems are polynomially solvable with some assumptions (Lawler, 1976; Lawler and Moore, 1969; Sahni, 1976). When some other details, such as set-up times, release times, precedence constraints, etc are added to the problem, the problem turns into an NP-hard problem (Steiner, 1997; Rote and Woeginger, 1998; Dauzère-Pérés, 1995; Moore, 1968; Baptiste et al., 2003; Dauzère-Pérés and Sevaux, 1999; Sevaux and Dauzere-Peres, 2003; Gordon and Kubiak, 1998; and Peridy et al., 2001).

Various exact, heuristic and metaheuristic methods were developed for single machine scheduling. Lawler and Moore (1969), Held and Karp (1962), Hariri and Potts (1994), and Kise et al. (1978) used dynamic programming (DP), while Villareal and Bulfin (1983), Potts and Wassenhove (1998), and Hallah and Bulfin (2003) used branch and bound algorithm (B&B). Yoo and Martin-Vega (2001) compared their hybridized genetic algorithm with other algorithms. There are also examples of bicriteria scheduling problems which are solved using B&B algorithm or GA (Güner et al., 1998; Chang and Su, 2001; and Köksalan and Keha, 2003). Among those articles, Lawler and Moore's DP approach can solve problems with up to 1000 jobs (1969), Potts and Wassenhove's B&B can solve problems with up to 100 jobs (1998), and Hallah and Bulfin's heuristic can solve problems with up to 2500 jobs (2003).

There are few articles (Gupta and Hariri, 1997; Croce et al., 2000; Lin, 2001; and Bulfin and Hallah, 2003) written about minimizing the number of tardy jobs on two machines, $F2 | \sum U_i$. The complexity of the problem is first settled by Lenstra et al. (1977) to be strongly NP-hard.

Scheduling n jobs on permutation flowshop problems are mainly concerned with the minimization of the completion time or maximum tardiness (Iyer et al., 2004; Reeves et al, 1998; Bertel and Billaut, 2004; Taillard, 1990; Ying and Liao, 2004; Tasgetiren et al., 2004 a and c; and Leu and Hwang, 2002).

Hariri and Potts (1989) present the first exact algorithm for the NP-hard $Fm | \sum U_i$ problem. The B&B algorithm they used can solve problems with up to 3 machines with 25 jobs. Lodree et al. (2004) propose a new sequencing rule, Earliest Adjusted Due Date (EADD) for the $Fm | r_j | \sum U_i$ problems with up to 50 jobs on 3 machines, and lastly Bertel and Billaut (2004) propose a GA to minimize the weighted number of tardy jobs in small sized flow shops with 8 jobs on 3 machines at most.

4. Genetic Algorithm

Genetic algorithms start with a population of solutions, whereas most stochastic search methods start with a single solution. An initial population is formed randomly or by means of a heuristic algorithm. Solutions are encoded in a form, which are called chromosomes. Each chromosome shows a complete solution to a problem. They are each assigned a fitness score that represents the ability of chromosomes to compete for mating and staying alive.

Parents are picked up to mate according to their fitness values. The fitter chromosomes produce more offspring than the less fit chromosomes. The solution set is then imposed to crossover, mutation and inversion. These stochastic operators are required for diversifying the solution pool and especially getting better solutions. Since the size of the population should be maintained statically, some weak individuals in the population die, and better solutions thrive to stay alive. The cycle continues until a certain number of iterations is executed or once the population converges. The solution procedure is summarized in the pseudo-code in Table 1.

Table 1 Pseudo-code of the Genetic Algorithm

```
Genetic Algorithm ()
{Initialize population  $P$  of size  $\lambda$            /* a randomly generated population*/
Evaluate  $\lambda$  individuals in  $P$            /*check the fitness of each chromosome*/
While termination criteria not satisfied do
{ Select  $2 * \mu$  individuals from  $P$ 
Crossover individuals to produce  $\mu$  offspring
Mutate some individuals in  $\mu$ 
Add  $\mu$  offspring to  $\lambda$  individuals in  $P$ 
Evaluate  $(\lambda + \mu)$  individuals in  $P$ 
Select  $\lambda$  individuals from  $(\lambda + \mu)$  individuals in  $P$  }
End Genetic Algorithm ()}
```

5. Particle Swarm Optimization Algorithm

PSO is a rather successful method for the continuous optimization problems; however, it is very difficult to adapt it for the discrete case; many studies have been done on it. These approaches can solve the combinatorial problems to some extent.

The PSO paradigm resembles to GA at some points. The initialization of the algorithm is done with a population of random solutions. New generations are formed by means of velocity updates. The optimal solution is searched among the updated generations. The potential solutions, called particles, fly through the multi-dimensional search space, and follow the current optimum particles. Execution of the algorithm is terminated as soon as the maximum number of iteration or maximum CPU time is exceeded.

All particles in the pool are kept during the whole run. PSO does not incorporate the survival of the fittest, whereas all other evolutionary algorithms do. Each particle has a velocity. Particles are carried to new positions with this velocity. The fitness values of particles are evaluated according to their positions at each iteration.

There is a communication between particles, each particle shares its information with others. A particle exchanges its information with the particles in the neighborhood. Therefore, after some number of iterations the swarm loses its diversity and the algorithm converges to the optimal solution.

In recent years, PSO has been successfully applied in many areas. It solves a variety of optimization problems in a faster and cheaper way than the evolutionary algorithms in the early iterations. It has few parameters to adjust and it works well for different kind of problems when the algorithm is slightly modified. The pseudo-code of the algorithm is given in Table 2.

Table 2 Pseudo-code of the PSO Algorithm

<p><i>Initialize parameters</i> <i>Initialize population</i> <i>Find permutation</i> <i>Evaluate</i> <i>Do</i> { <i>Find the personal best</i> <i>Find the global best</i> <i>Update velocity</i> <i>Update position</i> <i>Find permutation</i> <i>Evaluate</i> <i>Apply local search (optional)</i> } <i>While (Termination)</i></p>
--

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 represented by n number of dimensions as $X_i^t = [x_{i1}^t, x_{i2}^t, \dots, x_{in}^t]$, where x_{ij}^t is the position value of the i th particle with respect to the j th dimension ($j = 1, 2, \dots, n$).

Population: pop^t is the set of ρ particles in the swarm at iteration t , i.e., $pop^t = [X_1^t, X_2^t, \dots, X_\rho^t]$

Permutation: We introduce a new variable π_i^t , which is a permutation of jobs implied by the particle X_i^t . It can be described as $\pi_i^t = [\pi_{i1}^t, \pi_{i2}^t, \dots, \pi_{im}^t]$, where π_{ij}^t is the assignment of job j of the particle i in the permutation at iteration t .

Particle velocity: V_i^t is the velocity of particle i at iteration t . It can be defined as $V_i^t = [v_{i1}^t, v_{i2}^t, \dots, v_{im}^t]$, where v_{ij}^t is the velocity of particle i at iteration t with respect to the j th dimension.

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

Personal best: P_i^t represents the best position of the particle i with the best fitness until iteration t , 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, P_i^t can be determined and updated at each iteration t . In a minimization problem with the objective function $f(\pi_i^t)$ where π_i^t is the corresponding permutation of particle X_i^t , the personal best P_i^t of the i th

particle is obtained such that $f(\pi_i^t) \leq f(\pi_i^{t-1})$ where π_i^t is the corresponding permutation of personal best P_i^t and π_i^{t-1} is the corresponding permutation of personal best P_i^{t-1} . To simplify, we denote the fitness function of the personal best as $f_i^{pb} = f(\pi_i^t)$. For each particle, the personal best is defined as $P_i^t = [p_{i1}^t, p_{i2}^t, \dots, p_{in}^t]$ where p_{ij}^t is the position value of the i th personal best with respect to the j th dimension ($j = 1, 2, \dots, n$).

Global best: G^t 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 $f(\pi^t) \leq f(\pi_i^t)$ for $i = 1, 2, \dots, \rho$ where π^t is the corresponding permutation of global best G^t and π_i^t is the corresponding permutation of personal best P_i^t . To simplify, we denote the fitness function of the global best as $f^{gb} = f(\pi^t)$. The global best is then defined as $G^t = [g_1^t, g_2^t, \dots, g_n^t]$ where g_j^t is the position value of the global best with respect to the j th dimension ($j = 1, 2, \dots, n$).

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.

Solution Representation

In order to construct a direct relationship between the problem domain and the PSO particles for the PFSP, we present n number of dimensions for n number of jobs. In other words, each dimension represents a typical job. In addition, the particle $X_i^t = [x_{i1}^t, x_{i2}^t, \dots, x_{in}^t]$ corresponds to the continuous position values for n number of jobs in the PFSP. Table 3 illustrates the solution representation of particle X_i^t for the PSO algorithm with its corresponding permutation. According to the SPV rule in Tasgetiren et al. (2004 a, b, c, and d, 2005) the parameter vector is converted to the job permutation as shown in Table 3.

Table 3. Solution Representation of Particle X_i^t

Dimension, j	1	<u>2</u>	3	4	5	6
x_{ij}^t	1.80	<u>-0.99</u>	3.01	-0.72	-1.20	2.15
v_{ij}^t	3.89	2.94	3.08	-0.87	-0.20	3.16
Jobs, π_{ij}^t	5	<u>2</u>	4	1	6	3

This representation is unique in terms of finding new solutions since positions of each particle are updated at each iteration t in the PSO algorithm, thus resulting in different permutations at each iteration t .

Initial Population

A population of particles is constructed randomly. The continuous values of positions are established randomly. The following formula is used to construct the initial continuous position values of the particle uniformly:

$$x_{ij}^0 = x_{\min} + (x_{\max} - x_{\min}) * r_1$$

where $x_{\min} = -1.0$, $x_{\max} = 1.0$, and r_1 is a uniform random number between 0 and 1.

Initial velocities are generated by similar formula as follows:

$$v_{ij}^0 = v_{\min} + (v_{\max} - v_{\min}) * r_2$$

where $v_{\min} = -1.0, v_{\max} = 1.0$, and r_2 is a uniform random number between 0 and 1. Continuous velocity values are restricted to some range, namely

$$v_{ij}^t = [v_{\min}, v_{\max}] = [-1.0, 1.0]$$

where $v_{\min} = -v_{\max}$. Population size is the twice the number of dimensions. As the formulation of the problem suggests that the objective is to minimize the number of tardy jobs, the fitness function value is the number of tardy jobs for the particle i in the swarm. That is,

$$f_i^t(\pi_i^t) = c_{\min}(\pi_i^t, m)$$

where π_i^t is the corresponding permutation of particle X_i^t . For simplicity, $f_i^t(\pi_i^t)$ will be denoted as f_i^t .

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

Step 1: Initialization

Set $t=0, \rho$ =twice the number of dimensions.

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

Generate the initial velocities for particle i randomly, $\{V_i^0, i = 1, 2, \dots, \rho\}$ where $V_i^0 = [v_{i1}^0, v_{i2}^0, \dots, v_{in}^0]$.

Apply the SPV rule to find the permutation $\pi_i^0 = [\pi_{i1}^0, \pi_{i2}^0, \dots, \pi_{in}^0]$ of particle X_i^0 for $i = 1, 2, \dots, \rho$.

Evaluate each particle i in the swarm using the objective function f_i^0 for $i = 1, 2, \dots, \rho$.

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

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

Step 2: Update iteration counter

$$t = t + 1$$

Step 3: Update inertia weight

$$w^t = w^{t-1} * \beta \text{ where } \beta \text{ is decrement factor.}$$

Step 4: Update velocity

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

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

Step 5: Update position

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

Step 6: Find Permutation

Apply the SPV rule to find the permutation $\pi_i^t = [\pi_{i1}^t, \pi_{i2}^t, \dots, \pi_{in}^t]$ for $i = 1, 2, \dots, \rho$.

Step 7: Update personal best\

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

Step 8: Update global best

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

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

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.

6. Experimental Design

The $Fm | \sum U_i$ problem is known to be NP-hard; therefore, it is computationally expensive to solve the problem with exact algorithms such as B&B or DP. We have chosen the PSO algorithm to solve the problem and compared its performance with a traditional GA in terms of the execution time (cpu) and the number of tardy jobs.

The algorithms were coded in Borland C++ and runs were done on a Centrino 1.5 GHz computer with 256 MB memory. 10 replications of 20 instances were run for 20x15, 20x20, 30x15, 30x20, 40x15, 40x20, 50x15 and 50x20 problem sets (Demirkol et al., 1998).

The crossover probability of GA is tuned to 70 % and the mutation probability to 5 %. Initially, a population of size λ is constructed probabilistically which equals twice the number of jobs. At each generation, two parents with a random selection are determined to produce an offspring through order-based crossover and the crossover has been maintained until μ offspring are produced. The tournament selection of size 2 is used to establish the next population. By this way μ individuals are replaced among $\lambda + \mu$ individuals. The algorithm is run for 2500 generations.

Regarding the PSO parameters, social and cognitive parameters are taken as $c_1 = c_2 = 2$ consistent with the literature. Initial inertia weight is set to $w^0 = 1.2$ and is never decreased below 0.40. Finally, the decrement factor β is taken as 0.975. Population size is twice the number of dimensions. The SPV rule is used to convert a position vector to a job permutation. The obtained personal best and the global best values are recorded in the memory. The particles are mutated with a probability of 5 %. The algorithm converges after 2500 iterations.

7. Results

Table 4 is a comparison of the performances of GA and PSO algorithms in terms of number of tardy jobs. As seen, the average fitness values of PSO are less than those of GA. The cpu statistics are given in Table 5. PSO seems to be giving better results for the data sets 20x15, 20x20 and 30x15 problems. GA is faster to solve for more complex data sets.

Table 4 Comparison of fitness values

	GA	PSO	GA STD.	PSO STD.	GA	PSO	GA	PSO
J x M	AVG	AVG	DEV	DEV	MAX	MAX	MIN	MIN
20X15	17.21	16.88	2.53	2.71	20	20	10	10
20X20	18.37	18.24	2.13	2.34	20	20	13	13
30X15	23.26	22.8	4.04	4.16	29	29	14	14
30X20	25.44	25.03	4.2	4.29	30	30	16	16
40X15	27.27	26.87	6.38	6.17	36	36	15	15

40X20	31.17	30.85	5.4	5.27	39	39	20	20
50X15	31.9	31.56	7.76	7.32	44	43	17	18
50X20	35.99	35.58	7.78	7.38	47	46	21	21

Table 5 Comparison of cpu values

	GA	PSO	GA STD.	PSO STD.	GA	PSO	GA	PSO
J x M	AVG	AVG	DEV	DEV	MAX	MAX	MIN	MIN
20X15	1.46	0.97	0.6	0.05	5.54	1.08	1.02	0.9
20X20	1.71	1.07	0.59	0.05	5.07	1.17	1.11	1
30X15	2.51	1.86	0.95	0.08	9.15	2	1.66	1.69
30X20	1.93	2.1	0.07	0.23	2.39	4.37	1.8	1.94
40X15	2.62	3.22	0.09	0.03	2.8	3.33	2.45	3.11
40X20	2.96	3.64	0.05	0.03	3.11	3.72	2.8	3.58
50X15	3,69	5,14	0,08	0,04	4,26	5,27	3,45	5,01
50X20	4,23	5,81	0,08	0,05	4,5	6,06	3,9	5,68

We checked the normality of the data and did paired t-test to for all data sets to see which algorithm performed better. We established our hypothesis as;

$$H_0 : \mu_{GA} - \mu_{PSO} = 0$$

$$H_1 : \mu_{GA} - \mu_{PSO} > 0$$

In all data sets, we saw that the null hypothesis was rejected since all p-values obtained were less than the 5% significance level, which meant that PSO gave more promising results than GA.

Table 5 Paired t-test results

JxM	P-Value	JxM	P-Value
20x15	0,00000011	40x15	0,00000305
20x20	0,00005449	40x20	0,00000355
30x15	0,00000010	50x15	0,00068857
30x20	0,00000009	50x20	0,00001573

8. Conclusions

To the best of our knowledge, our study stands to be the first to apply the particle swarm optimization algorithm to minimization of tardy jobs problem in permutation flow shops. We enabled that algorithm to be applied to the discrete case by using a simple SPV heuristic rule. Then, we compared its performance with a traditional GA that we developed. We saw that the PSO algorithm gave very promising results for finding better sequences.

It will be more meaningful if we generate the due dates which are more realistic. It is also among our further studies to develop more meaningful upper and lower bounds for due dates. Ours is still one of the important studies and it will be more valuable if we polish it with further research.

References

- Angeline, P. Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Difference. the 7th Annual Conference on Evolutionary Programming. San Diego. USA. 1998.
- Baptiste. P., Peridy, L. Pinson. E. A branch and bound to minimize the number of late jobs on a single machine with release time constraints. *European Journal of Operational Research*. Vol. 144. pp: 1–11. 2003.
- Bean. J. C.. Genetic Algorithm and Random Keys for Sequencing and Optimization. *ORSA Journal on Computing*. Vol. 6 (2). pp: 154-160. 1994
- Beasley. D.. Bull D. R.. Martin. R. R.. An Overview of Genetic Algorithms: Part 1. Fundamentals. *University Computing*. Vol. 15 (2). pp: 58 -69. 1993.
- Beasley. D.. Bull D. R.. Martin. R. R.. An Overview of Genetic Algorithms: Part 2. Research Topics. *University Computing*. Vol. 15 (4). pp: 170-181. 1993
- Bertel. S.. Billaut. J.. C.. A genetic algorithm for an industrial multiprocessor flowshop scheduling problem with recirculation. *European Journal of Operational Research*. Vol. 159. pp: 651–662. 2004.
- Bulfin. R.L.. Hallah. R.. Minimizing the weighted number of tardy jobs on a two-machine flow shop. *Computers & Operations Research*. Vol. 30. pp: 1887–1900. 2003.
- Carlisle. A.. and Dozier. G.. Adapting Particle Swarm Optimization to dynamic environments. WAC 2002 Proceedings. Orlando. Florida. June 9-13. 2002
- Carlisle. A.. and Dozier. G.. An Off-the-Shelf PSO. Proceedings of the Workshop on Particle Swarm Optimization. Indianapolis. In: Purdue School of Engineering & Technology. 2001.
- Chang. P.. Su. L.. Scheduling n Jobs on One Machine to Minimize the Maximum Lateness with Minimum Number of Tardy Jobs. *Computers & Industrial Engineering*. Vol. 40. pp: 349-360. 2001
- Clerc. M.. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. Proceedings. 1999 ICEC. Washington. DC. pp 1951-1957. 1999.
- Croce. F. D.. Gupta. J.. N.. D.. Roberto Tadei. Minimizing tardy jobs in a flowshop with common due date. *European Journal of Operational Research*. Vol.120. pp: 375-381. 2000.
- Dauzère-Pérés. S.. Minimizing late jobs in the general one machine scheduling problem. *European Journal of Operational Research*. Vol. 81 pp: 134–142. 1995.
- Dauzère-Pérés. S.. Sevaux. M.. A Branch and Bound Method to Minimize the Number of Late Jobs on a Single Machine In National contribution for the 15th triennial conference. IFORS'99. Beijing. P.R. of China. 16-20 August 1999
- Dauzère-Pérés. S.. Sevaux. M.. Using Lagrangean Relaxation to Minimize the (Weighted) Number of Late Jobs on a Single Machine. 1999.
- Demirkol E.. Mehta S. and Uzsoy R.. Benchmarks for shop scheduling problems. *European Journal of Operational Research*. Vol. 109. pp: 137-141. 1998
- Eberhart. R. C.. and Kennedy. J.. A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micro Machine and Human Science. Nagoya. Japan. Piscataway. NJ: IEEE Service Center. pp: 39-43. 1995.
- Eberhart. R. C.. and Shi. Y.. Comparison between Genetic Algorithms and Particle Swarm Optimization. In V. W. Porto. N. Saravanan. D. Waagen. A. E. Eiben. Eds. *Evolutionary Programming VII: Proc. Seventh Ann. Conf. on Evolutionary Programming Conf.*. San Diego. CA. Berlin: Springer-Verlag. 2001.

- Eberhart, R. C., and Shi, Y., Particle Swarm Optimization: Developments, Applications and Resources. Proc. congress on evolutionary computation 2001 IEEE service center. Piscataway, NJ., Seoul, Korea., 2001
- Eberhart, R.C., and Shi, Y., Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. Congress on Evolutionary Computing. Vol. 1. pp. 84-88. 2000
- Gordon, V., Kubiak, W., Single Machine Scheduling with Release and Due Date Assignment to Minimize the Weighted Number of Tardy Jobs. Information Processing Letters. Vol. 68. pp: 153-159. 1998.
- Güner, E., Erol, S., Tani, K., One machine scheduling to minimize the maximum earliness with minimum number of tardy jobs. Int. J. Production Economics. Vol. 55. pp: 213-219. 1998.
- Gupta, J. N. D., Hariri, A.M.A., Two-machine flowshop scheduling to minimize the number of tardy jobs. Journal of the Operational Research Society. Vol. 48. pp: 212–220. 1997
- Hallah R., M., Bulfin, R., L., Minimizing the weighted number of tardy jobs on a single machine. European Journal of Operational Research. Vol. 145. pp: 45-56. 2003
- Hariri, A.M.A., Potts, C.N., A Branch and Bound Algorithm to Minimize the Number of Late Jobs in a Permutation Flowshop. European Journal of Operational Research. Vol. 38 pp: 228-237. 1989
- Hariri, A.M.A., Potts, C.N., Single machine scheduling with deadlines to minimize the weighted number of tardy jobs. Management Science. Vol. 40. pp: 1712-1719. 1994
- Held, M., Karp, R. M., A Dynamic Programming Approach to Sequencing Problems. Journal of the Society for Industrial and Applied Mathematics. Vol. 10(1). pp: 196-210. 1962
- Hino, C. M., Ronconi, D. P., Mendes A. B., Minimizing earliness and tardiness penalties in a single-machine problem with a common due date. European Journal of Operational Research. Vol. 160. pp: 190–201. 2005
- Holland, J., Adaptation in natural and artificial system. Ann Arbor, MI: The University of Michigan Press. 1975
- Iyer, S., K., Saxena, B., Improved Genetic Algorithm for the Permutation Flowshop Scheduling Problem. Computers & Operations Research. Vol. 31. pp: 593–606. 2004
- Karp, R., M. Reducibility Among Combinatorial Problems., In: R. E. Miller and J. W. Thatcher, Eds., Complexity of Computer Computations. pp. 85.103. Plenum Press. New York, NY, USA. 1972.
- Kennedy, J. Eberhart, R. C., Shi, Y., Swarm Intelligence. San Francisco: Morgan Kaufmann Publishers. 2001
- Kennedy, J., and Eberhart, R. C., Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks. IV. 1942-1948. Piscataway, NJ: IEEE Service Center. 1995
- Kennedy, J., Eberhart, R., Particle Swarm Optimization. IEEE International Conference on Neural Networks. IEEE Service Center. Piscataway, NJ. Volume: 4. pp. 1942-1948. 1995
- Kethley, R., B., Alidaee, B., Single machine scheduling to minimize total weighted late work: a comparison of scheduling rules and search algorithms. Computers & Industrial Engineering. Vol. 43. pp: 509–528. 2002
- Kise, H., Ibaraki, T., Mine, H. A solvable case of the one machine scheduling problem with ready and due times. Operations Research. Vol. 26 (1). pp: 121–126. 1978
- Köksalan, M., Keha, A. B., Using genetic algorithms for single-machine bicriteria scheduling problems. European Journal of Operational Research. Vol. 145. pp: 543–556. 2003
- Lawler, E. L., Moore, C. U., A Functional Equation and its Application to Resource Allocation and Sequencing Problems. Management Science. Vol. 16. pp. 77.84. 1969

- Lawler. E. L. Sequencing to minimize the weighted number of tardy jobs. *RAIRO Rech. Optr.* Vol. 10. pp: 27-33. 1976.
- Lawler. E. L.. A Dynamic Programming Algorithm for Preemptive Scheduling of a Single Machine to Minimize the Number of Late Jobs. *Annals of Operations Research.* Vol. 26. pp: 125-133. 1990
- Lenstra. J.K.. Rinnooy Kan. A.H.G.. Bruker. P.. Complexity of machine scheduling problems. *Annals of Discrete Mathematics.* Vol. 1. pp: 343-362. 1977
- Leu. S.. Hwang. S.. GA-based resource-constrained flow-shop scheduling model for mixed precast production. *Automation in Construction.* Vol. 11. pp: 439– 452. 2002.
- Lin. B.M.T.. Cheng. T.C.E.. Minimizing the weighted number of tardy jobs and maximum tardiness in relocation problem with due date constraints. *European Journal of Operational Research.* Vol. 116. pp: 183-193. 1999
- Lin. B.M.T.. Scheduling in the two-machine flowshop with due date constraints *Int. J. Production Economics.* Vol. 70. pp: 117-123. 2001
- Lodree. E.. Jang. W.. Klein. C.. M.. A New Rule for Minimizing the Number of Tardy Jobs in Dynamic Flowshops. *European Journal of Operational Research.* Vol. 159. pp: 258-263. 2004
- Moore. J. M.. A n Job. One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs. *Management Science.* Vol. 15. No. 1. pp. 102.109. 1968.
- Pinedo. M.. *Scheduling: Theory. Algorithms. and Systems.* Englewood Cliffs. N.J.: Prentice Hall. 1995
- Potts. C.N.. Wassenhove. L. N.. Algorithms for scheduling a single machine to minimize the weighted number of late jobs. *Management Science* Vol. 34. pp: 843–858. 1988.
- Rote. G.. Woeginger. G. J.. Minimizing the Number of Tardy Jobs on a Single Machine with Batch Setup Times. *START Project Y43-MAT Combinatorial Approximation Algorithms.* 1998
- Sahni. S. K.. Algorithms for Scheduling Independent Jobs. *J. Assoc. Comput. Mach.* Vol. 23. pp: 116-127. 1976
- Sevaux. M.. Dauzere-Peres. S.. Genetic algorithms to Minimize the Weighted Number of Late Jobs on a Single Machine. *European Journal of Operational Research.* Vol. 151. pp: 296-306. 2003
- Shi. Y.. Eberhart. R. C.. Parameter selection in particle swarm optimization. *Evolutionary Programming VII. Lecture Notes in Computer Science.* vol. 1447. Springer. pp. 591–600. 1998
- Shi. Y.. Eberhart. R.C.. A modified particle swarm optimizer. *Proceedings of the IEEE International Conference on Evolutionary Computation.* pp. 303–308. 1998
- Sipper. D.. Bulfin. R. L.. *Production: Planning. Control and Integration.* McGraw Hill. 1998
- Steiner. G.. Minimizing the number of tardy jobs with precedence constraints and agreeable due dates. *Discrete Applied Mathematics.* Vol. 72. pp: 167- 177. 1997
- Taillard. E.. Some Efficient Heuristic Methods for the FlowShop Sequencing Problem. *European Journal of Operational Research.* Vol. 47. pp: 65–74. 1990
- Tasgetiren M. F.. Sevkli M.. Yun-Chia Liang. Gencyilmaz G. Particle Swarm Optimization Algorithm for the Single Machine Total Weighted Tardiness Problem. *World Congress on Evolutionary Computation. CEC2004.*p.1412-1419. 2004.
- Tasgetiren. M. F. Liang Y. C.. A Binary Particle Swarm Optimization Algorithm for Lot Sizing Problem. *Journal of Economic and Social Research.* Vol.5 No.2. 2003.
- Tasgetiren. M. F.. Liang Y. C.. Sevkli M.. Yenisey. M. M.. Particle Swarm Optimization and Differential Evolution Algorithms for Job Shop Scheduling. 2005 (submitted)

- Tasgetiren. M. F.. Liang Y. C.. Sevkli. M.. Gencyilmaz. G.. Particle Swarm Optimization Algorithm for Permutation Flowshop Sequencing Problem. 4th International Workshop on Ant Colony Optimization and Swarm Intelligence. ANTS2004. LNCS 3172 by Springer-Verlag. pp.382-390. September 5-8. 2004
- Tasgetiren. M. F.. Liang Y. C.. Sevkli. M.. Gencyilmaz. G.. Particle Swarm Optimization Algorithm for Makespan and Total Flowtime Minimization in Permutation Flowshop Sequencing Problem. European Journal of Operational Research. 2004
- Tasgetiren. M. F.. Liang Y. C.. Sevkli. M.. Gencyilmaz. G.. Particle Swarm Optimization Algorithm for Makespan and Maximum Lateness Minimization in Permutation Flowshop Sequencing Problem. 4th International Symposium on Intelligent Manufacturing Systems. IMS2004. pp.431-441. Sakarya. Turkey. 6-8 Sep 2004
- Tasgetiren. M. F.. Sevkli. M.. Liang Y. C.. Gencyilmaz. G.. Particle Swarm Optimization Algorithm for Single Machine Total Weighted Tardiness Problem. 4th International Symposium on Intelligent Manufacturing Systems. IMS2004. pp.431-441. Sakarya. Turkey 6-8 Sep 2004
- Trelea. I. C.. The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection. Information Processing Letters. Vol. 85. pp: 317-325. 2003
- Villareal. F.. J.. Bulfin. R.. L.. Scheduling a single machine to minimize the weighted number of tardy jobs. IIE Transactions. Vol 15. pp: 337-343. 1983
- Yoo. W. S.. Martin-Vega. L. A.. Scheduling Single Machine Problems for On-time Delivery. Computers & Industrial Engineering. Vol. 39. pp: 371-392. 2001

