Using Hybrid Particle Swarm Optimization to solve Machine Time Scheduling Problem with Random Starting Time

M. A. Shohla[1], A. A. El-sawy[2], M. Nofal[3] and S. F. El-Zoghdy[4]

[1,3] Computer Engineering Dept., Faculty of Computers and Information Technology, Taif University Taif, KSA.

[2,4] Computer Science Dept., Faculty of Computers and Information Technology, Taif University Taif, KSA

Abstract

The starting time in the machine time scheduling problem will be assumed stochastic follows certain distribution. A hybrid algorithm combines the mutation operation with particle swarm optimization algorithm with constriction factor has been developed to find best starting time for each machine in each cycle when starting time follows normal distribution.

Keywords: *Machine Time Scheduling, Particle Swarm optimization, Mutation, Time Window, Normal Distribution.*

1. Introduction

A great deal of research has been focused on solving scheduling problems. One of the most important scheduling problems is the Machine Time Scheduling Problem (MTSP). This problem was investigated in [16] as a parameterized version of the MTSP, which was defined in [10], with penalized earliness in starting and lateness in the completion of the operation. The authors in [16] applied the optimal choice concept which is given in [14] and some theoretical results from [15] to obtain the optimal values of the given parameters.

In [3] the authors investigated two cycles MTSP and introduced an algorithm to find the optimal choice of parameters, which represent the earliest possible starting time for the second cycle. In [2] an algorithm was developed (MTSP Algorithm (MTSPA)) for multi-cycles MTSP which found the starting time for each machine in each cycle by using the max-separable technique. The processing times in the previous researches were deterministic. A generalization was introduced in [1] to overstep the cases at which an empty feasible set of solutions is described by the system.

In [4] introduced an algorithm by using the PSO and GA to solve MTSP, and compared between PSO, GA and max-separable technique (using numerical example). The Authors found that PSO algorithm reach to the best solution than GA and max-separable technique. In [12] discusses how to solve the MTSP when the processing time for each machine is stochastic. To solve this problem, the Monte Carlo simulation is suggested to handle the given stochastic processing times. In this paper we will introduce an algorithm to find best starting time for each machine in each cycle when starting time is stochastic. We assumed that the starting time is random variable with follows normal distribution. The starting time bounded with time window, so when we reformulated this tome window we will keep on the randomness distribution for the starting time. We will develop a hybrid algorithm by using particle swarm optimization algorithm (with constriction factor) with the mutation operation to solve the machine time scheduling problem when starting time in stochastic case.

2. Problem Formulation

In machine time scheduling problem there are n machines, each machine carries out one operation j with processing time p_j for $j \in N = \{1,...,n\}$ and the machines work in k cycles. Let $\tilde{x}_{jr}$ represent starting time of the j^{th} machine in cycle r for all $j \in N$, $r \in K = \{1,...,k\}$ (k number of cycles) and is random variable with certain distribution. Machine j can start its work in cycle r only after the machines in a given set $N^{(j)}$, $N^{(j)} \subset N$ ($N^{(j)}$ is the set of precedence machines) had finished their work in the $(r\text{-}1)^{th}$ cycle, so we can define the starting time in the $(r\text{-}1)^{th}$ cycle as follows:

$$\tilde{x}_{ir+1} \geq \max_{j \in N^{(i)}} (x_{jr} + p_{jr}) \quad \forall i \in N, \forall r \in K \qquad (1)$$

Assuming that the starting time x_{jr} is constrained by a time interval $[l_{jr}, L_{jr}]$ for each $j \in N$, $r \in K$ and, then the set of feasible starting times $\tilde{x}_{jr}$ is described by the following system for each $r \in K$:

$$\max_{j\in N^{(i)}} (\tilde{x}_{jr} + p_{jr}) \le \tilde{x}_{ir+1} \quad \forall i \in N,$$
$$l_{jr} \le \tilde{x}_{jr} \le L_{jr} \qquad \forall j \in N \tag{2}$$

Assume also that for some echological reasons there are a given recommended time interval $[a_{jr}, b_{jr}]$, $\forall i \in N$, $\forall r \in K$ so:

$$[\tilde{x}_{jr}, \tilde{x}_{jr} + p_j] \subset [a_{jr}, b_{jr}], \tag{3}$$

The violation of the Eq. (3) will be penalized by the following penalty function

$$f(\tilde{x}) = \max_{j\in N} f_{jr}(\tilde{x}_{jr}) \to \min \qquad r \in K \tag{4}$$

Where the penalty function in a certain cycle r is given by:

$$f_{jr}(\tilde{x}_{jr}) = \max\{ f_{jr}^{(1)}(\tilde{x}_{jr}), f_{jr}^{(2)}(\tilde{x}_{jr} + p_{jr}), 0\} \quad \forall j \in N$$

Where $f_{jr}^{(1)}$: R **à** R is decreased continuous function such that $f_{jr}^{(1)}(a_{jr})=0$,

And $f_{jr}^{(2)}$: R **à** R is increasing continuous function such that $f_{jr}^{(2)}(b_{jr})=0$

To minimize the maximum penalty in each cycle r, we should solve the following problem:

$$f(\tilde{x}) \to \min$$

subject to :

$$\max_{j\in N^{(i)}} (\tilde{x}_{jr} + p_{jr}) \le \tilde{x}_{ir+1} \quad \forall i \in N \tag{5}$$
$$l_{jr} \le \tilde{x}_{jr} \le L_{jr} \qquad \forall j \in N$$

3. Particle Swarm Optimization

The PSO method is a member of wide category of Swarm Intelligence methods for solving the optimization problems. It is a population based search algorithm where each individual is referred to as particle and represents a candidate solution. Each particle in PSO flies through the search space with an adaptable velocity that is dynamically modified according to its own flying experience and also the flying experience of the other particles. Further, each particle has a memory and hence it is capable of remembering the best position in the search space ever visited by it. The position corresponding to the best fitness is known as *pbest* and the overall best out of all the particles in the population is called *gbes*t [13].

The modified velocity and position of each particle can be calculated using the current velocity and the distance from the $pbest_j$ to *gbest* as shown in the following formulas:

$$v_{j,g}^{(t+1)} = w * v_{j,g}^{(t)} + c_1 * r_1 * (pbest_{j,g} - x_{j,g}^{(t)}) + c_2 * r_2 * (gbest_{j,g} - x_{j,g}^{(t)}) \tag{6}$$

$$x_{j,g}^{(t+1)} = x_{j,g}^{(t)} + v_{j,g}^{(t+1)} \tag{7}$$

With $j=1, 2, \ldots, n$ and $g=1, 2, \ldots, m$

n =number of particles in a group;

m = number of members in a particle;

t = number of iterations (generations);

$v_{j,g}^{(t)}$ =velocity of particle j at iteration t,

w = inertia weight factor;

c_1, c_2 = cognitive and social acceleration factors, respectively;

r_1, r_2 = random numbers uniformly distributed in the range (0, 1);

$x_{j,g}^{(t)}$ = current position of j at iteration t;

$pbest_j$ = *pbest* of particle j;

gbest = *gbest* of the group.

The index of best particle among all of the particles in the group is represented by the *gbest.* In PSO, each particle moves in the search space with a velocity according to its own previous best solution and its group's previous best solution. The velocity update in a PSO consists of three parts; namely momentum, cognitive and social parts. The balance among these parts determines the performance of a PSO algorithm. The parameters c_1 & c_2 determine the relative pull of *pbest* and *gbest* and the parameters r_1 & r_2 help in stochastically varying these pulls [13]. [5] Showed that combining them by setting the inertia weight, c , to the constriction factor, v, improved performance across a wide range of problems as follows:

$$v_{j,g}^{(t+1)} = c\{w * v_{j,g}^{(t)} + c_1 * r_1 * (pbest_{j,g} - x_{j,g}^{(t)}) + c_2 * r_2 * (gbest_{j,g} - x_{j,g}^{(t)})\} \tag{8}$$

$$c = \frac{2}{\left|2 - c - \sqrt{c^2 - 4c}\right|} \quad where\ c = c_1 + c_2,\ c < 4. \tag{9}$$

In [6] propose a simple modification to the particle swarm algorithm in which, at each generation, a small number of particles are mutated and are allowed to hill climb. The mutation has the effect of randomly bouncing particles towards other parts of the search space, while the effect of hill-climbing is to greatly increase the effective size of the "target" region of interest around the global optimum. We provide results for a selection of well-known test functions, and demonstrate that our modification improves the ability of the swarm to find the global optimum.

[8] Presented a new mutation operator called the Systematic Mutation (SM) operator for enhancing the performance of Basic Particle Swarm Optimization (BPSO) algorithm. The SM operator unlike most of its contemporary mutation operators do not use the random probability distribution for perturbing the swarm population, but uses a quasi random Sobol sequence to find new solution vectors in the search domain. The

comparison of SM-PSO is made with BPSO and some other variants of PSO. The empirical results show that SM operator significantly improves the performance of PSO.

4. Mutation

GA maintains a set of candidate solutions called population and repeatedly modifies them. At each step, the GA selects individuals from the current population to be parents and uses them produce the children for the next generation. Candidate solutions are usually represented as strings of fixed length, called chromosomes. A fitness or objective function is used to reflect the goodness of each member of population [12]. Two operators in GA have been used to generate next generation called crossover and mutation.

In mutation, a bit involves flipping it: changing a 0 to 1 or vice versa. The parameter P_m (the mutation rate), gives the probability that a bit will be flipped. The bits of a string are independently mutated that is, the mutation of a bit does not affect the probability of mutation of other bits. For example, suppose all the strings in a population have converged to a 0 at a given position and the optimal solution has a 1 at that position. Then crossover cannot regenerate a 1 at that position, while a mutation could [9].

5. Normal Distribution

The normal distribution described in [7] as the most important one in all of probability and statistics. Many numerical populations have distribution that can be fit much closed by an appropriate normal curve Fig 1. A random variable X is said to have a normal distribution with parameter m and s where $-\infty < x < \infty$ and $s > 0$ if the probability density function:

$$f(x;m,s) = \frac{1}{\sqrt{2p}\,s} e^{-(x-m)^2/(2s^2)} \qquad -\infty < x < \infty \qquad (10)$$

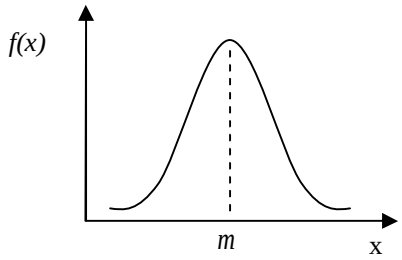

Fig. 1 Probability Density Function for Normal Distribution

To compute $p(a \le x \le b)$ when x is normal random variable with parameter m and s , we must evaluate

$$\int_a^b \frac{1}{\sqrt{2p}\,s} e^{-(x-m)^2/(2s^2)} dx \qquad (11)$$

None of the standard integration techniques can be used to evaluate previous expression. For $m = 0$ and $s = 1$, previous expression has been numerically evaluated and tabulated for certain values of *a* and *b* [7]. This table can also be used to compute probabilities for any other values of m and s .The normal distribution with parameter $m = 0$ and $s = 1$ is called standard normal distribution. A random variable that has a standard normal distribution is called standard normal random variable and denoted by *z:*

$$z = \frac{x-m}{s} \qquad (12)$$

With probability density function:

$$f(z;m,s) = \frac{1}{\sqrt{2p}} e^{-z^2/2} \qquad -\infty < x < \infty \qquad (13)$$

So to find the values of *x* random variable with normal distribution with any parameter m and s by $x = m + s\ z$

Where z could be calculated without need to the stander normal tables with the expression, $Z = (\cos 2pR_2)\sqrt{-2\log R_1}$ [11] where R_1 and R_2 are two independent random numbers between zero and one.

From the standard normal probability table we found that about 95% of all values fall within 2 standard deviations of the mean, that is $P(m-2s \le x \le m+2s) = 0.9544$ and about 99% all values fall within 3 standard deviations of the mean, that is $P(m-3s \le x \le m+3s) = 0.9974$.

6. Solving stochastic MSTP by HPSOM-MTSP algorithm

Now, we will develop an algorithm to solve machine time scheduling problem where the starting time for each machine in each cycle is stochastic with certain normal distribution by using hybrid particle swarm optimization algorithm with construction factor. The new algorithm will be called HPSOM-SMTSP, follow the next steps:

1- Boundaries Determination:

If the machine starting time follows uniform distribution then the L_{ir} and l_{ir} equal to the uniform boundaries.

If the machine starting time follows normal distribution then

$l_{ir} = m_{ir} - 3s_{ir}$, $L_{ir} = m_{ir} + 3s_{ir}$ $i \in n, r \in K$

2- Reformulation:

Each machine boundaries will be reformulate (calculate the new boundaries) based on its' successors

machines boundaries. For each machine the new lower boundary called *h* and the new upper boundary called *H*.

3- New Normal Parameters:

If machine starting time normal distribution then we will find the new m'_{ir}, s'_{ir} $i \in n, r \in K$ by solving the following equations together:

$h_{ir} = m'_{ir} - 3s'_{ir}$,

$H_{ir} = m'_{ir} + 3s'_{ir}$ $i \in n, r \in K$

Note: we keep the shape of normality

4- Initial iteration:

First, the particle defines as a set of starting times for the machines in all cycles. The particle is represented by *D*-dimensional, where *D* equal to *N* multiplies by *K* (where *N* number of machine and *K* number of cycles). The $\tilde{x}_{irpt}$ is the starting time for machine *i* in cycle *r* in particle *p*, $p = 1,2,..,Q$ in iteration *t*, $t = 1,2,..,T$ (where *Q* is number of particles in the swarm and *T* is number of iterations) which satisfy the constraints in (P). If $\tilde{x}_{irpt}$ is normal distribution then $x_{irpt} = m'_{ir} + zs'_{ir}$ where $Z = (\cos 2pR_2)\sqrt{-2\log R_1}$. The x_{irpt} must satisfy the second constrain which is $\tilde{x}_{irpt} \geq \max_{j \in N^{(i)}} (\tilde{x}_{j(r-1)pt} + p_j)$. Determine the $pbest_p$ which is the best position of particle *p* that make the best value of the objective function. Then determine the *gbest* which is the best particle that make the best value of the objective function in all iterations.

5- Other iterations:

The next iteration created by modifying the velocity of each particle by the following equation:

$$v_{j,g}^{(t+1)} = c\{w * v_{j,g}^{(t)} + c_1 * r_1 * (pbest_{j,g} - x_{j,g}^{(t)}) + c_2 * r_2 * (gbest_{j,g} - x_{j,g}^{(t)})\}$$

$$c = \frac{2}{\left|2 - c - \sqrt{c^2 - 4c}\right|}$$

where $c = c_1 + c_2$, $c < 4$.

Then the particle position will be update by the following equation:

$$\tilde{x}_{irp(t+1)} = \tilde{x}_{irpt} + v_{irp(t+1)}$$

6- Mutation:

The symbol *A* denoted that the number of particle will be mutated. And the symbol *E* is the number of members in the particle that will be mutated. If x_{irat} $a \in A$ is normal distributed then the value of these members will be generated randomly based on new m'_{ir}, s'_{ir}.

7- Calculate the objective function then find the $pbest_p$ and *gbest*

8- *Repeat* step 5 to step 7 until the *T*.

HPSOM-SMTSP Algorithm:

A1: If x_{irpt} normal distribution then

$l_{ir} = m_{ir} - 3s_{ir}$, $L_{ir} = m_{ir} + 3s_{ir}$ $i \in n, r \in K$

A2: Reformulate the boundaries for each machine in each cycle as follows:

Put $H_{ik} = L_{ik}$ $\forall i \in N$, $h_{jr} = l_{jr}$ $\forall j \in N$,

$H_{jr} = \min(L_{jr}, (\min_{i \in U_j} H_{ir+1} - p_j))$ Where

$U_j = \{i \in N : j \in N^{(i)}\}$ $r = k-1,...,2,1$

A3: Put *t* = 1.

A4: Put *p* = 1.

A5: Put *r* = 1.

A6: Put *i* = 1.

A7: If $r \neq 1$ then $h_{ir} = \max_{j \in N^{(i)}} (\tilde{x}_{j(r-1)pt} + p_j)$.

A8: If $\tilde{x}_{irpt}$ normal distribution then finds the new m'_{ir}, s'_{ir} by solving the following equations together:

$h_{ir} = m'_{ir} - 3s'_{ir}$,

$H_{ir} = m'_{ir} + 3s'_{ir}$ $i \in n, r \in K$

then find $\tilde{x}_{irpt} = m'_{ir} + s'_{ir} z$ where

$Z = (\cos 2pR_2)\sqrt{-2\log R_1}$ else generate random number for $\tilde{x}_{irpt}$ where $h_{ir} \leq \tilde{x}_{ircp} \leq H_{ir}$.

A9: If $i < n$ then $i = i + 1$ go to A7.

A10: If $r < k$ then $r = r + 1$ go to A6.

A11: $pbest_p = f(\tilde{x}_{irpt})$ $\exists i = 1,..,N$, $\exists r = 1,..,K$.

A12: If $p < Q$ then $p = p + 1$ go to A5.

A13: find $\min(f(pbest_p))$ $\exists p = 1,..,Q$.

A14: $gbest = pbest_{p_{\min}}$

A15: $t = t + 1$.

A16: Put $p = 1$.

A17: $v_{j,g}^{(t+1)} = c\{w * v_{j,g}^{(t)} + c_1 * r_1 * (pbest_{j,g} - x_{j,g}^{(t)}) + c_2 * r_2 * (gbest_{j,g} - x_{j,g}^{(t)})\}$

$c = \frac{2}{\left|2 - c - \sqrt{c^2 - 4c}\right|}$ *where* $c = c_1 + c_2$, $c < 4$.

A18: $\tilde{x}_{irpt} = \tilde{x}_{irp(t-1)} + v_{irpt}$.

A19: if $\tilde{x}_{irpt}$ is not feasible then go to A21.

A20: if $f(\tilde{x}_{irpt}) > f(\tilde{x}_{irpt-1})$ then $pbest_p = \tilde{x}_{irpt}$
A21: If $p < Q$ then $p = p + 1$ go to A17.
A22: if $f(gbest) > f(pbest_{p_{\min}})$ then $gbest = pbest_{p_{\min}}$.
A23: $f(gbest_t) < f(gbest_{t-1})$ then go to 16.
A24: Generate random number A between 1 and Q.
A25: a=1.
A26: Put $e = 1$.
A27: Generate random number m between 1 and D.
A28: Put $r = trunc(m/N) + 1$.
A29: Put $i = rem(m/N)$.
A30: if x_{irat} normal distributed then $\tilde{x}_{irat} = m'_{ir} + s'_{ir} z$ where $Z = (\cos 2pR_2)\sqrt{-2\log R_1}$ else generate random number for $\tilde{x}_{irpt}$ where $h_{ir} \le \tilde{x}_{ircp} \le H_{ir}$.
A31: if $e < E$ then $e = e + 1$ go to A27.
A32: if $a < A$ then $a = a + 1$ go to A28.
A33: if $f(gbest) > f(\tilde{x}_{ir1t})$ then $gbest = \tilde{x}_{ir1t}$.
A34: If $t < T$ then go to A15.
A35: The solution is *gbest.*

trunc(): function return the integer part of divided operation.
rem(): function return the remaining value divided operation.

7. Numerical Example

Consider a problem with the following values of parameters n = 5 so $N = \{1,2,3,4,5\}$, and processing time $p = \{2,4.5,6.25,4,5\}$. The starting time for machine 2 in cycle 1 belongs to normal distribution with $m = 2$, $s = 2/3$ and machine 3 in cycle 2 belongs to normal distribution with $m = 6$, $s = 0.25$. Other machines in other cycles belong to uniform distribution with boundaries as follows:

Table 1: Machine Boundaries

Cycle (r)	*r = 1*	*r = 2*	*r = 3*
s_{ir} i=1,2,…,5	{1,-,0,3,1}	{4,6,-,5,6}	{10,11,12,9,11.5}
S_{ir} i=1,2,…,5	{5,-,3,5,6}	{6.5,7,-,7.25,6.5}	{13,12,15,12,14}

Respect to machine which belongs to normal distribution we will find a boundaries for starting times for these machines to can reformulate all machines boundaries based on the their predecessors. So, most values for starting time x_{21} of machine 2 in cycle 1 (99% of values) fall in l_{21}, L_{21}. These boundaries calculated as follows:

$l_{21} = m_{21} - 3s_{21}$, $L_{21} = m_{21} + 3s_{21}$ then $l_{21} = 0$, $L_{21} = 4$.

So, most values for starting time x_{32} of machine 3 in cycle 2 (99% of values) fall in l_{32}, L_{32}. These boundaries calculated as follows:

$l_{32} = m_{32} - 3s_{32}$, $L_{32} = m_{32} + 3s_{32}$ then $l_{32} = 6$, $L_{32} = 7.5$.

Table 2: Calculated Machine Boundaries

Cycle (r)	*r = 1*	*r = 2*	*r = 3*
l_{ir} i=1,2,…,5	{1,0,0,3,1}	{4,6,6,5,6}	{10,11,12,9,11.5}
L_{ir} i=1,2,…,5	{5,4,3,5,6}	{6.5,7,7.5,7.25,6.5}	{13,12,15,12,14}

Table 3: Machine Relations

i	*1*	*2*	*3*	*4*	*5*
$N^{(i)}$	{1,2,3}	{2}	{2,3}	{1,4,5}	{1,3,5}
U_j	{1,4,5}	{1,2,3}	{1,3,5}	{4}	{4,5}

Assume further that
$f_{jr}(x_{jr}) = \max(a_{jr} - x_{jr}, x_{jr} + p_{jr} - b_{jr}, 0) \quad \forall j \in N$
Where a_j, b_j are for all $j \in N$ given constants so that we have in our case for all $j \in N$
$f_{jr}^{(1)}(x_{jr}) = a_{jr} - x_{jr}$ $\quad$ $f_{jr}^{(2)}(x_{jr} + p_{jr}) = x_{jr} + p_{jr} - b_{jr}$

Input values of a_{ir} and b_{ir} for each cycle

Table 4: Machines Penalty Boundaries

Cycle (r)	*r=1*	*r=2*	*r=3*
a_{ir} i=1,2,…,5	{1,1,1,3,3}	{5,7,6,5,7}	{11,12,11,10,13}
b_{ir} i=1,2,…,5	{4,6,8,5,5}	{8,9,8,6.5,8}	{13,15,14,12,14}

After Reformulation of the problem will obtain the following new boundary vectors:

Table 5: Reformulated Machine Boundaries

Cycle (r)	*r = 1*	*r = 2*	*r = 3*
h_{ir} i=1,2,…,5	{1,0,0,3,1}	{4,6,6,5,6}	{10,11,12,9, 11.5}
H_{ir} i=1,2,…,5	{4.5,2,0.25,3.25, 1.5}	{6.5,7,7.5,7.25, 6.5}	{13,12,15,12, 14}

We will keep on the normality shape of x_{21} and x_{32}. So, we will calculate the new m, s called m', s' based on the new boundaries, as follows:

$h_{21} = m'_{21} - 3s'_{21}$, $H_{21} = m'_{21} + 3s'_{21}$

$0 = m'_{21} - 3s'_{21}$, $2 = m'_{21} + 3s'_{21}$

$so, m'_{21} = 1$ *and* $s'_{21} = 1/3$

$h_{32} = m'_{32} - 3s'_{32}$, $H_{32} = m'_{32} + 3s'_{32}$

$6 = m'_{32} - 3s'_{32}$, $7.5 = m'_{32} + 3s'_{32}$

$so, m'_{32} = 6.75$ *and* $s'_{32} = 0.25$

We applied the HPSOM-SMTSP algorithm on this example. We found that, the best parameters for the algorithm are swarm size equal 80, the value of w equal 0.5 and the value c1 and c2 equal 1.7. We run the program 100 trials. The mutation probability equal 10% of particle size and the number of particles which will be mutated equal 20% of swarm size. We found that the best value of *averageF* equal 32.90 after 300 iterations as we show in the Fig. 2.

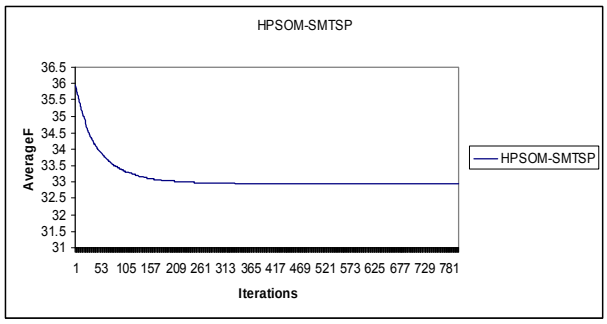

Fig. 2 The solution of HPSOM-SMTSP

The best starting time for each machine in each cycle is:

	M_1	M_2	M_3	M_4	M_5
C_1	1.7	1	0.12	3	1.44
C_2	6.37	6	6.37	7	6.43
C_3	12.62	11.24	12.62	11.43	12.62

8. Conclusion

An algorithm has been developed for solving machine time scheduling problem when starting time is random and follows certain distribution. We assumed that the starting time follows normal distribution. The new algorithm combines the mutation operation with particle swarm optimization algorithm with construction factor to develop a hybrid algorithm.

References

[1] A. Tharwat and A. Abuel-Yazid: "Generalized Algorithm For Muulti-Cycle Machine Time Scheduling", Proceeding of Meatip3 Conference, Assuit, Egypt, 2002.

[2] A. Tharwat and A. Abuel-Yazid: "Multi-Cycles Machine Time Scheduling Problem", First International Conference on Informatics and Systems, Cairo, Egypt, 2002.

[3] A. Tharwat and K. Zimmermann: "Optimal Choice of Parameters in Machine Time Scheduling Problems Case I," Conference MMEI, Liberc, Czech Republic, 1998.

[4] A. A. Sawy and A. A. Tharwat, " Comparison of Particle SWARM Optimization, Genetic Algorithm and Max separable Technique for Machine Time Scheduling Problem ", 5th international Conference on Mathematics and Engineering Physics, May 2010.

[5] Alec Banks, Jonathan Vincent, Chukwudi Anyakoha "A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications" NATURAL COMPUTING Volume 7, Number 1, 109-124.

[6] I. H. Grundy and A. Stacey, "Particle swarm optimization with combined mutation and hill climbing" Proceedings of 7th Asia-Pacific Conference on Complex Systems 2004 (Complex 2004), Cairns, Australia.

[7] J. L. Devore "Probability distribution and statistics for Engineering and Sciences". Fourth Edition, 1995, Wadsworth Inc.

[8] Millie Pant, Radha Thangaraj, V. P. Singh and Ajith Abraham, "Particle Swarm Optimization Using Sobol Mutation", International Conference on Emerging Trends in Engineering and Technology, ICETET 2008, IEEE Computer Society Press, USA, ISBN 978-0-7695-3267-7, pp. 367-372, 2008.

[9] M. Srinivas, Lalit M. Patnaik "Genetic Algorithms: A Survey", IEEE 1994.

[10] M. Vlach and K. Zimmermann, "Machine Time Scheduling Synchronization of Starting Times", the Proceeding of the MME'99 Conference, Prague, Czech Republic, 1999.

[11] R. A. Gagliano,"Simulation By The Monte Carlo Process", Technical Paper, Georgia Tech, 1974.

[12] S. A. Hassan, A.A. Tharwat, I.A. El-Khodary, A. A. El-Sawy "Using Monte Carlo Simulation to Solve Machine Time Scheduling Problems With Stochastic Processing Time", Mathematical Methods in Economics conference: MME'2003, Prague , Czech Republic , 10-12 Sept.

[13] S. Panda and N. P. Padhy, "Comparison of Particle Swarm Optimization and Genetic Algorithm for TCSC-based Controller Design", International Journal of Computer Science and Engineering, Volume 1 Number 1, 2007

[14] S. Zlobec: "Input Optimization I: Optimal Realizations of Mathematical Models," Mathematical Programming, V 31, pp.245-268, 1985.

[15] S. Zlobec: "Input Optimization III: Optimal Realizations of Mathematical Models," Mathematical Programming, V 17, No 4, pp.429-445, 1986.

[16] Y. Sok and K. Zimmermann: "Optimal Choice of Parameters in Machine Time Scheduling Problems with Penalized Earliness in Starting Time and Lateness", AUC-Mathematica et Physica, V33, No 1, pp.53-61. 1992.