The Solution of Machines' Time Scheduling Problem Using Artificial Intelligence Approaches

Ghoniemy S.
Computer Engineering Dept.,
Faculty of Computers and Information Technology, Taif University
Taif, KSA.

El-sawy A. A.
Computer Science Dept.,
Faculty of Computers and Information Technology, Taif University
Taif, KSA.

Shohla M. A.
Computer Engineering Dept.,
Faculty of Computers and Information Technology, Taif University
Taif, KSA.

Gihan E. H. Ali
Mathematics Dept.,
Faculty of Science and Education, Khurma Branch, Taif University,
Taif, KSA.

***Abstract*—The solution of the Machines' Time Scheduling Problem (MTSP) is a hot point of research that is not yet matured, and needs further work. This paper presents two algorithms for the solution of the Machines' Time Scheduling Problem that leads to the best starting time for each machine in each cycle. The first algorithm is genetic-based (GA) (with non-uniform mutation), and the second one is based on particle swarm optimization (PSO) (with constriction factor). A comparative analysis between both algorithms is carried out. It was found that particle swarm optimization gives better penalty cost than GA algorithm and max-separable technique, regarding best starting time for each machine in each cycle.**

Keywords- Machine Time Scheduling; Particle swarm optimization; Genetic Algorithm; Time Window.

I. 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 [1] as a parameterized version of the MTSP, which was defined in [2], with penalized earliness in starting and lateness in the completion of the operation. The authors in [1] applied the optimal choice concept which is given in [3] and some theoretical results from [4] to obtain the optimal values of the given parameters.

In [5] 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 [6] an algorithm was developed (MTSP Algorithm (MTSPA)) for multi-cycle 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. The authors in [7] discussed how to solve the MTSP when the processing time for each machine is stochastic. To solve this problem, the Monte Carlo simulation was suggested to handle the given stochastic processing times. A generalization was introduced in [8] to overstep the cases at which an empty feasible set of solutions is described by the system.

This paper examines two approaches for solving the MTSP; PSO (with constriction factor), and GA (with non-uniform mutation). A comparative analysis between both algorithms is to be carried out for the solution that minimizes the penalty cost regarding the best starting time for each machine in each cycle.

The paper is organized as follows. Part 2 formulates the problem. Part 3 introduces the proposed GA, as well as its implementation. Part 4 presents the PSO Algorithm, and its implementation. Both algorithms, in addition to the max-separable technique, are applied for a specific scenario. Obtained results are investigated in Part 5. The paper is terminated by conclusions and proposals for future work.

II. PROBLEM FORMULATION

In machines' 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 xjr 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). 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 $(r-1)^{th}$ cycle, so we can define the starting time in the $(r+1)^{th}$ cycle as follows:

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

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

$$\max_{j \in N^{(i)}} (x_{jr} + p_{jr}) \le x_{ir+1} \quad \forall i \in N,$$

$$l_{jr} \le x_{jr} \le L_{jr} \qquad \forall j \in N \qquad (1)$$

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

$$[x_{jr}, x_{jr} + p_j] \subset [a_{jr}, b_{jr}], \qquad (2)$$

The violation of (2) will be penalized by the following penalty function

$$f(x) = \max_{j \in N} f_{jr}(x_{jr}) \rightarrow \min \qquad r \in K$$

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

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

where $f_{jr}^{(1)}$: R→ R is a decreasing continuous function such that $f_{jr}^{(1)}(a_{jr}) = 0$,

and $f_{jr}^{(2)}$: R→ R is an 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:

$$f(x) \rightarrow \min$$

$$subject\ to:$$

$$\max_{j \in N^{(i)}} (x_{jr} + p_{jr}) \le x_{ir+1} \quad \forall i \in N \qquad (3)$$

$$l_{jr} \le x_{jr} \le L_{jr} \qquad \forall j \in N$$

III. Proposed Approaches For Solving The MTSP

A. Using Genetic Algorithm (GA):

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 to 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 [9]. The principle of genetic algorithms is simple [10]:

- Encoding of the problem in a binary string.
- Random generation of a population. This one includes a genetic pool representing a group of possible solutions.
- Reckoning of a fitness value for each subject. It will directly depend on the distance to the optimum.
- Selection of the subjects that will mate according to their share in the population global fitness.
- Genomes crossover and mutations.
- And then start again from point 3.

This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

Non-uniform mutation has been used to reduce the disadvantage of random mutation in the real-coded GA [11]. This new operator is defined as follows. For each individual X_i^t in a population of tth generation, create an offspring X_i^{t+1} through non-uniform mutation as follows: if $X_i^t = \{x_1, x_2, ..., x_m\}$ is a chromosome (t is the generation number) and the element xk is selected for this mutation, the result is a vector $X_i^{t+1} = \{x'_1, x'_2, ..., x'_m\}$ where

$$x'_k = \begin{cases} x_k + \Delta(t, UB - x_k) & if\ a\ random\ \xi\ is\ 0. \\ x_k - \Delta(t, x_k - LB) & if\ a\ random\ \xi\ is\ 1. \end{cases} \qquad (4)$$

and LB and UB are the lower and upper bounds of the variables x_k. The function $\Delta(t, y)$ returns a value in the range [0,y] such that $\Delta(t, y)$ approaches to zero as t increases. This property causes this operator to search the space uniformly initially (when t is small), and very locally at later stages. This strategy increases the probability of generating a new number close to its successor than a random choice. We use the following function:

$$\Delta(t, y) = y.(1 - r^{(1 - \frac{t}{T})^b}). \qquad (5)$$

Where r is a uniform random number from [0, 1], T is the maximal generation number, and b is a system parameter determining the degree of dependency on the iteration number.

In [12], the authors introduced a new mutation operator characterized by its non-uniformness. The operator has been used in the parameter optimization of the controllers of a supply ship. Four different kinds of controllers have been considered and optimized, providing a wide range of optimization problems with their own unique search spaces to test the mutation operator.

The main steps for solving the MTSP using GA are as follows:

1) Reformulation:

Each machine boundaries will be reformulated (calculate the new boundaries) based on its' successors machines boundaries. For each machine, the new lower boundary is called h and the new upper boundary is called H.

2) Initial population:

First, the chromosome is defined as a set of starting times for the machines in all cycles. So, S is the size of the chromosome is equal to n multiplied by k (n number of machines and k number of cycles). The gene x_{ircp} is the starting times for machine i in cycle r in chromosome c in population p

(Q number of chromosomes and W number of population) which satisfy the constraint in (P_r). The value of x_{ircp} is generated randomly where $h_{ir} \le x_{ircp} \le H_{ir}$.

3) Other generation:

The value of fitness for each chromosome in the previous population had been calculated. The next generation is created by selecting the best chromosomes which have the greatest value in the objective function. T is a percent of the previous population which determines the number of best chromosomes that transfer to the next generation.

4) Crossover:

The remaining chromosomes are divided as a pair. Each chromosome in each pair is divided in certain cycle v then swap between v to k cycle in first chromosome and v to k cycles in the second chromosome.

5) Mutation:

The chromosome will be mutated based on the non-uniform mutation; equation (4), and equation (5).

The steps form 3 to 5 will be repeated until the last number of population. The solution is the first chromosome of the last population.

6) GAnuM-MTSP Algorithm:

GA1: Reformulate the boundaries for each machine in each cycle as follows:

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

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

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

GA2: Put $p = 1$.
GA3: Put $c = 1$.
GA4: Put $r = 1$.
GA5: Put $i = 1$.
GA6: If $r \neq 1$ then $h_{ir+1} = \max_{j \in N^{(i)}} (x_{ircp} + p_{jr}) \quad \forall i \in N$
GA7: Generate random number for x_{ircp} where $h_{ir} \le x_{ircp} \le H_{ir}$.
GA8: If i < n then i = i + 1 go to GA6.
GA9: If r < k then r = r + 1 go to GA5.
GA10: If c < Q then c = c + 1 go to GA4.
GA11: $x_{irc(p+1)} = x_{ircp} \quad \forall i \in n, \forall r \in k, \forall c \in Q.$
GA12: Sort descending $f_{cp}(x_{cp})$ if $p = W$ then go to GA26.
GA13: Put $c = Q - (T*Q)$.
GA14: Put $r = v$.
GA15: Put $i = 1$.
GA16: Swap between x_{ircp} and $x_{ir(c+1)p}$.
GA17: If $i < n$ then $i = i + 1$ go to GA16.
GA18: If $r < k$ then $r = r + 1$ go to GA15.
GA19: If $c < Q$ then c = c + 2 go to GA14.
GA20: Put $c = 1$.
GA21: Put $r = 1$.
GA22: Put $i = 1$.
GA23: Generate random number *rand* between 0 and 1.
GA24: *if* $rand \ge 0.5$ *then* $x_{irap} = x_{irap} + (H_{ir} - x_{irap}).(1 - r^{(1-\frac{p}{w})^b})$
GA25: if $rand < 0.5$ then $x_{irat} = x_{irat} - (x_{irat} - h_{ir}).(1 - r^{(1-\frac{t}{T})^b})$
GA26: If $i < n$ then $i = i + 1$ go to GA23.
GA27: If $r < k$ then $r = r + 1$ go to GA22.
GA28: If $c < Q$ then c = c + 1 go to GA21.
GA29: If $p < W$ then $p = p + 1$ go to GA3.
GA30: The solution is $x_{ir1p} \quad \forall i \in n, \forall r \in k.$

7) Simulation Results:

Consider the MTSP with the following parameters:

n = 5, i.e. $N = \{1,2,3,4,5\}$, $p = \{2,4.5,6.25,4,5\}$, the machines boundaries are as shown in Table I, and the machines relations are as shown in Table II. 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 aj, bj are for all $j \in N$ given constants so that we have in our case for all $j \in N$

$$f_{jr}^{(1)}(x_{jr}) = a_{jr} - x_{jr} \qquad f_{jr}^{(2)}(x_{jr} + p_{jr}) = x_{jr} + p_{jr} - b_{jr}$$

Input values of a_{ir} and b_{ir} for each cycle are as shown in Table I.

TABLE I. 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 II. 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}

TABLE III. 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}

The GA-MTSP algorithm is implemented and run for solving the allocated problem on a computer with processor Intel Centrino 1.6 GHz with 215 MB RAM. The population size has been tested by 40, 60, 80 and 100 chromosomes. The result shows that, the best population size is 100 chromosome as shown in figure (1-a). The number of chromosomes that will be kept in the next population has been tested by 30%, 20%,

10%, 5%, 2.5% and 1.25% of population size. It was found that, the best number of chromosomes that will be kept in the next population equals 1.25% from population size as shown in figure (1-b). This means that the best chromosome, which has the best fitness function, will be transferred to the next population. The rest of chromosomes will be crossovered together to generate the rest of next population. It means that the probability of crossover is 98.75%, which is another parameter of GA. Using single point crossover, it was found that the best position of cutting point is 33% of the chromosome size as shown in figure (1-c). Finally, the Genetic algorithm parameters that give the best starting times have been determined:

TABLE IV. MACHINE STARTING TIME BY GANUM-MTSP

	M1	M2	M3	M4	M5
C1	1.18	0.85	0.07	3.1	1.36
C2	6.49	6.07	6.67	7.3	6.48
C3	13	11.27	13.13	11.5	13.3

The best value for the objective function f is 35.26. Assuming that α is the cost, and then the penalty cost equals 35.26 α

B. Using Particle Swarm Optimization (PSO)

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 gbest [9].

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:

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

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

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;

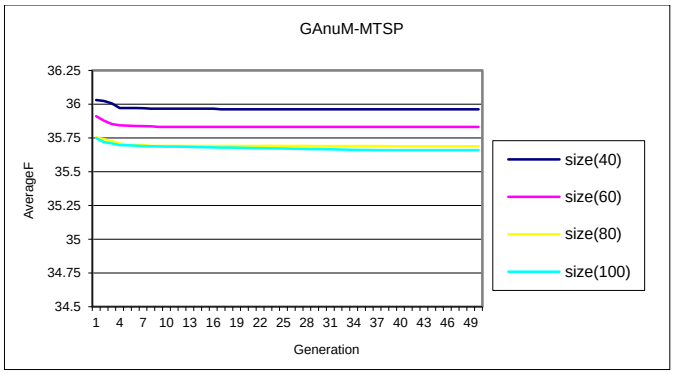
c_1, c_2 = cognitive and social acceleration factors, respectively;

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

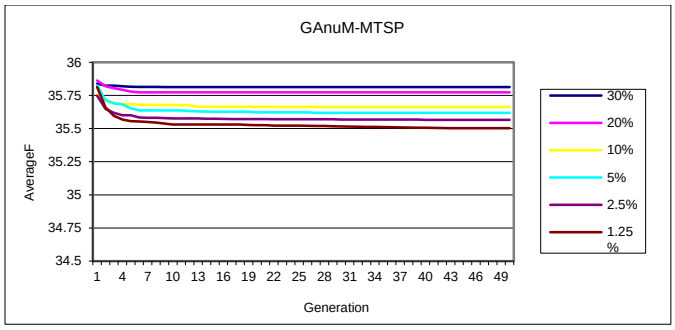
$x_{j,g}^{(t)}$ = current position of j at iteration t;

$pbest_j$ = *pbest* of particle j;

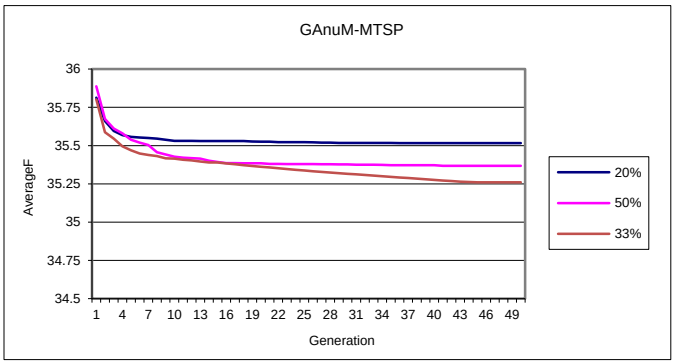
gbest = *gbest* of the group.



a) Population size



b) No. of chromosomes kept in next generation



c) Position of the cutting point crossover

Figure 1. Genetic algorithm parameters

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 [9]. [13] Showed that combining them by setting the inertia weight, χ , to the constriction factor, v, improved performance across a wide range of problems as follows:

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

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

In [14] PSO is combined with the Lagrangian Relaxation (LR) framework to solve a power-generator scheduling problem known as the unit commitment problem (UCP). In terms of the solution quality, the PSO-LR provided a "best solution" with a lower cost than GA for problem sizes larger than 20 units, and than LR for problem sizes 20 and 80 units. PSO-LR also provided a "best solution", for the problem size of 10 units, with a much lower cost than using PSO alone.

A novel approach based on Particle Swarm Optimization (PSO) for scheduling jobs on computational grids is introduced in [15]. The proposed approach is to dynamically generate an optimal schedule so as to complete the tasks within a minimum period of time as well as utilizing the resources in an efficient way. When compared to genetic algorithm (GA) and simulating Annealing (SA), an important advantage of the PSO algorithm is its speed of convergence and the ability to obtain faster and feasible schedules.

Application and performance comparison of PSO and GA optimization techniques were presented in [9], for Thyristor Controlled Series Compensator (TCSC)-based controller design. Results indicate that in terms of computational time, the GA approach is faster. The computational time increases linearly with the number of generations for GA, whereas for PSO the computational time increases almost exponentially with the number of generations. The higher computational time for PSO is due to the communication between the particles after each generation. However, the PSO seems to arrive at its final parameter values in fewer generations than the GA.

The main steps for solving the MTSP by PSO are as follows:

1) Reformulation:

Each machine boundaries will be reformulated (calculate the new boundaries) based on its' successors machines boundaries. For each machine the new lower boundary is called h and the new upper boundary is called H.

2) Initial iteration:

First, the particle is defined as a set of starting times for the machines in all cycles. The particle is represented by *D*-dimensional, where *D* is equal to *N* multiplied by *K* (where *N* number of machines and *K* number of cycles). The 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). The value of x_{irpt} is generated randomly where $h_{ir} \le x_{irpt} \le H_{ir}$.

The x_{irpt} must satisfy the second constrain which is $x_{irpt} \ge \max_{j \in N^{(i)}} (x_{j(r-1)pt} + p_j)$. Determine the $pbest_p$ which is the best position of particle *p* that makes 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.

3) Other generation:

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

$$v_{irp(t+1)} = \chi\{w * v_{irpt} + c_1 * r_1 * (pbest_{irp} - x_{irpt}) + c_2 * r_2 * (gbest_{ir} - x_{irpt})\}$$

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

$$x_{irp(t+1)} = x_{irpt} + v_{irp(t+1)}$$

The new iteration has been created with new position of SWARM. Calculate the objective function then find the $pbest_p$ and gbest. Repeat this step until last iterations. The solution is the gbest in the last iteration.

4) PSOc-MTSP Algorithm:

A1: Reformulate the boundaries for each machine in each cycle as follows:

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

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

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

A2: Put *t* = 1.
A3: Put *p* = 1.
A4: Put *r* = 1.
A5: Put *i* = 1.
A6: If $r \neq 1$ then $h_{ir} = \max_{j \in N^{(i)}} (x_{j(r-1)pt} + p_j)$.
A7: Generate random number for x_{ircp} where $h_{ir} \le x_{ircp} \le H_{ir}$.
A8: If $i < n$ then $i = i + 1$ go to A6.
A9: If $r < k$ then $r = r + 1$ go to A5.
A10: $pbest_p = f(x_{irpt})_{pt}$ $\exists i = 1,..,N, \exists r = 1,..,K$.
A11: If $p < Q$ then $p = p + 1$ go to A4.
A12: find max $(f(pbest_p)_{pt})$ $\exists p = 1,..,Q$.
A13: $gbest = pbest_{p_{\max}}$
A14: $t = t + 1$.
A15: Put $p = 1$.
A16: $v_{irpt} = \chi\{w * v_{irp(t-1)} + c_1 * r_1 * (pbest_p - x_{irp(t-1)}) + c_2 * r_2 * (gbest - x_{irp(t-1)})\}$
.
A17: $x_{irpt} = x_{irp(t-1)} + v_{irpt}$.
A18: if x_{irpt} is not feasible then go to A20.
A19: if $f(x_{irpt})_{pt} > f(x_{irpt})_{p(t-1)}$ then $pbest_p = x_{irpt}$
A20: $gbest = pbest_{p_{\max}}$.
A21: If $p < Q$ then $p = p + 1$ go to A16.
A22: If $t < T$ then go to A15.
A23: The solution is *gbest.*

---

5) Simulation Results

The PSOc-MTSP algorithm has been implemented and run for the same scenario used in Part 3.1.2. The program is run 100 times for determining the suitable parameters. Using test

sizes of 5, 20, 40 and 50 particle in the swarm, we found that, the best swarm size equals 20 as show in figure (2-a). After testing the w value by 0.1, 0.5, 0.9 and 0.9→ 0.1 (decreasing value) we found that the best value of w equals 0.5 as show in figure (2-b). The last parameters, we need to determine, are c_1, c_2. Using the values 0.5, 1, 1.5, 1.7, 1.9 and 2, for c1 and c2 we found that the best value for c_1, c_2 equal 1.5 as shown in figure (2-c). Finally, the swarm parameters that give the best starting times have been determined is shown below.

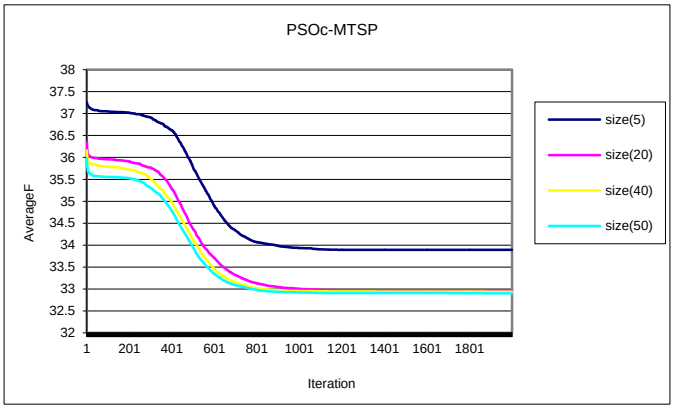
TABLE V. MACHINE STARTING TIME BY PSOC-MTSP

	M1	M2	M3	M4	M5
C1	1.95	1.25	0.0	3	1.26
C2	6.24	6	6.26	7	6.23
C3	12.5	11.26	12.5	11.27	12.5

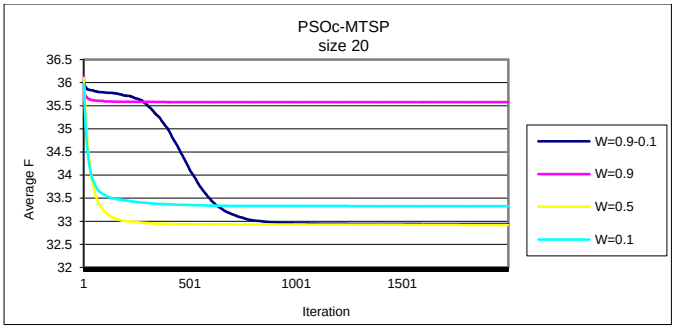
The best value for the objective function f equals 32.9. Assuming that the cost equal α. Then the penalty cost equals 32.9 α.

C. Discussion

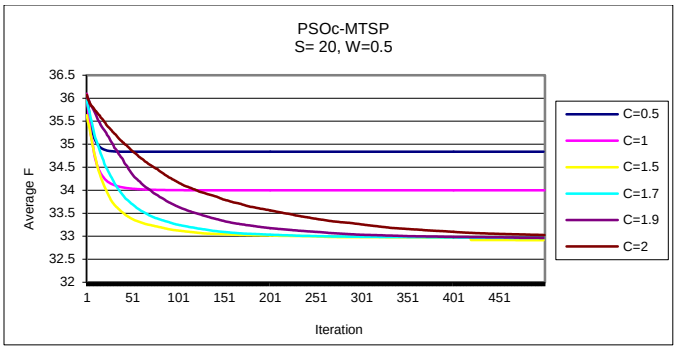
From previous experimental results we found that, solving the MTSP using particle swarm optimization algorithm (PSOc-MTSP), leads to 32.9α penalty cost in 420 iterations that took 11 seconds. When solving the MTSP using genetic algorithm (GAnuM-MTSP), the penalty cost was 35.25 α, reached in generation 41 that took 2 seconds. But when the MTSP was solved using max-separable algorithm in [2], the penalty cost was 35.75 α in less than 0.5 seconds, (Figure 3).



a) swarm size



b) *w* value



c) c_1, c_2 value

Figure 2. Determination of the swarm parameters

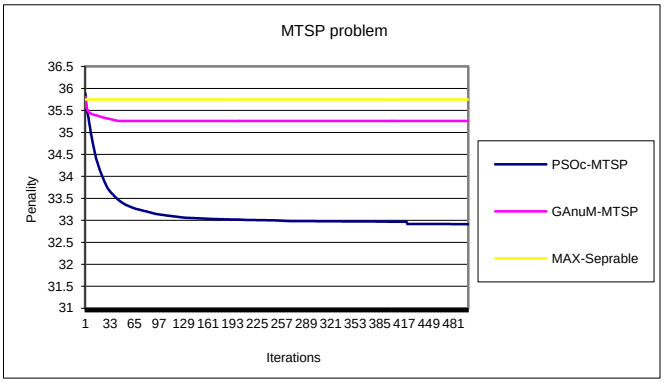


Figure 3. The result of solving MTSP problem by SWARM, GA and Max-separable

D. Conclusion

The machine time scheduling problem (MTSP) was solved using particle swarm optimization (with constriction factor), genetic algorithm (GA) (with non-uniform mutation), and max-separable technique. We found that, particle SWARM optimization gives the lowest penalty cost of the MTSP problem, followed by GA algorithm. The max-separable technique gives the highest penalty cost. That means that particle swarm optimization algorithm is the most suitable for solving the MTSP problem, giving the best starting time for each machine in each cycle.

REFERENCES

[1] [1] 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.

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

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

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

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

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

[7] 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.

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

[9] 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.

[10] Jean-Philippe Rennard, "Genetic Algorithm Viewer: Demonstration of a Genetic Algorithm", Ph.D. May 2000.

[11] Zhao X. and Gao X-S., "Evolutionary Programming Based on Non-Uniform Mutation", MM Research Preprints, MMRC, AMSS, Academia Sinica, No. 23, December 2004, Pages: 352-374.

[12] Alfaro-Cid E., McGookin E. W., Murray-Smith D. J., "A Novel Non-uniform Mutation Operator and its Application to the Problem of Optimizing Controller Parameters", IEEE, Vol. 2, 2005,pages: 1555 - 1562.

[13] 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.

[14] H. H. BALCI and J. F. VALENZUELA,"scheduling electric power generators using particle swarm optimization combined with the lagrangian relaxation method", Int. J. Appl. Math. Comput. Sci., Vol. 14, No. 3, 411–421, 2004.

[15] H. Liu, A. Abraham and C. Grosan, "A Novel Variable Neighborhood Particle Swarm Optimization for Multi-objective Flexible Job-shop Scheduling Problems", IEEE International Conference on Digital Information Management, Lyon, France, IEEE Press, USA, ISBN 1-4244-1476-8, pp. 138-145, 2007.