Military Technical College
Kobry Elkobbah,
Cairo, Egypt
May 25-27,2010

5th International Conference on
Mathematics and Engineering
Physics (ICMEP-5)

EM-30

Comparison of Particle SWARM Optimization, Genetic Algorithm and Max separable Technique for Machine Time Scheduling Problem

A. A. El-sawy **A. A. Tharwat**

Abstract

In this paper we deal with a multi cycle machine time scheduling problem (MTSP) to find the best starting time for each machine in each cycle. We introduce an algorithm by using the particle SWARM optimization (PSO) and Genetic algorithm to solve the MTSP. A comparison between PSO, GA and max-separable technique will be introduced to find the best solution which is the best starting time respect to its time window for each machine in each cycle and respect to the set of precedence machines to minimize the penalty cost.

KEYWORDS: Machine Time Scheduling, Particle SWARM optimization, Genetic Algorithm, Max-separable, Time Window**.**

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 [12] as a parameterized version of the MTSP, which was defined in [7], with penalized earliness in starting and lateness in the completion of the operation. The authors in [12] applied the optimal choice concept which is given in [10] and some theoretical results from [11] 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. In [8] 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. A generalization was introduced in [1] to overstep the cases at which an empty feasible set of solutions is described by the system. In this paper we will introduce an algorithm by using the PSO and GA to solve MTSP. Then, we will compare between PSO, GA and MTSPA (using numerical example) to find the best solution of starting time for each machine in each cycle that minimize the penalty cost.

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 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). 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:

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

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

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

$$l_{jr} \leq x_{jr} \leq L_{jr} \qquad \forall j \in N \qquad (I)$$

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:

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

The violation of the (*) 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 decreased continuous function such that $f_{jr}^{(1)}(a_{jr}) = 0$,

And $f_{jr}^{(2)}$: R→ 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:

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

subject to :

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

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

3- Max-separable Technique:

The authors in [2] used the max-separable technique for solving a multi-cycles MTSP problem (*P*) and the introduced algorithm can summarized as follows:

Step 1: Boundaries Reformulation; the new boundaries will be *h, H*.

Step 2: Put *r = 1*.

Step 3: Find the feasible set of starting times for the first cycle.

Step 4: Find the optimal set of starting times.

Step 5: Find the optimal starting times in the r^{th} Cycle based on $(r + 1)^{th}$ Cycle Intervals.

Step 6: Decrease the distance between boundaries in the $(r + 1)^{th}$ Cycle.

Step 7: Let *r = r + 1*, if $r \leq k - 1$ then returns to Step 3. Otherwise continue.

Step8: Find the feasible set of starting times for the k^{th} Cycle using the calculated values in the $(k\text{-}1)^{th}$ Cycle.

Step9: Find the optimal starting times in the k^{th} Cycle then Stop.

Now we will introduce a numerical example that explains the how the max-separable technique solves the MTSP as follows:

The example:
Consider a problem with the following values of parameters n = 5 so N = {1,2,3,4,5}, p = {2,4.5,6.25,4,5},

Table (1) 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 (2) 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 $f_{jr}(x_{jr}) = \max(a_{jr} - x_{jr}, x_{jr} + p_{jr} - b_{jr}, 0) \qquad \forall j \in N$

Where a_j, b_j are for all $j \in N$ given constants so that we have in our case for all $j \in N$

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

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

Table (3) 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}

By solving this example by max-separable technique, the results show that, the value of f equal 35.75, and the starting time for each machine in each cycle as follows:

	M_1	M_2	M_3	M_4	M_5
C_1	[1,2]	[1,1.5]	0.25	3	1.5
C_2	6.5	6	6.5	7	6.5
C_3	12.75	12	12.75	11.5	13

4- 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 *gbes*t [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;
c1 , *c2* = cognitive and social acceleration factors, respectively;
r1 , *r2* = 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 *c1 & c2* determine the relative pull of *pbest* and *gbest* and the parameters *r1 & r2* help in stochastically varying these pulls [9].
In [4] PSO 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 introduced in [5]. 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.

PSO Algorithm for solving MTSP

The main steps of the algorithm that solves MTSP by PSO as follows:

1- 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*.

2- 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 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} 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 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.

3- Other generation:

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

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

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.

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

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: $pbest_p = f(x_{irpt})_{pt} \quad \exists i = 1,..,N, \exists r = 1,..,K$.

A11: If $p < Q$ then $p = p + 1$ go to A4.

A12: find $\max(f(pbest_p)_{pt}) \quad \exists p = 1,..,Q.$

A13: $gbest = pbest_{p_{\max}}$

A14: $t = t + 1$.

A15: Put $p = 1$.

A16: $v_{irpt} = w * v_{irp(t-1)} + c_1 * r_1 * (pbest_p - x_{irp(t-1)}) + c_2 * r_2 * (gbest - x_{irp(t-1)})$.

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.

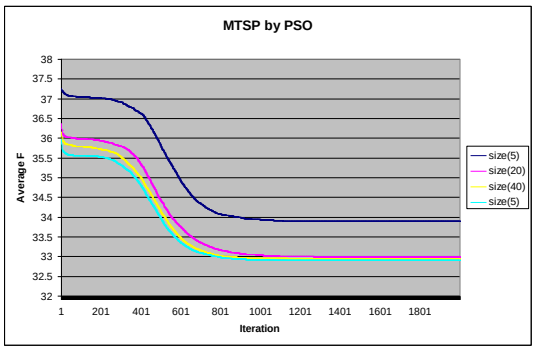
A22: If $t < T$ then go to A15.

A23: The solution is *gbest*.

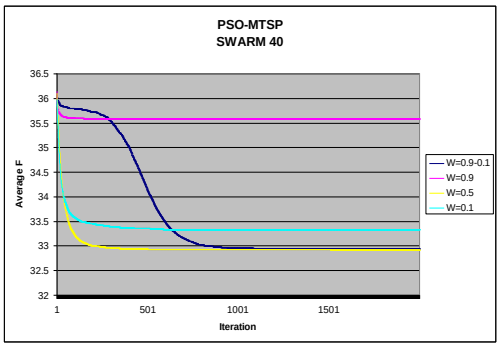
The example:

The PSO-MTSP algorithm has been applied on the previous example. We run the program 100 trials. We found that, the best swarm size equal 40 after we test the size by 5, 20, 40 and 50 particle in the swarm as show in figure (1-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* equal 0.5 as show in figure (1-b). The last parameters, we need to determine, are c_1, c_2. After tries the c_1, c_2 values by 0.5, 1, 1.5, 1.7, 1.9 and 2, we found that the best value for c_1, c_2 equal 1.7 as show in the figure (1-c). Finally, the swarm parameters have been determined that gives the best value for the objective function *f* (equal 32.96286) and the best starting times is:

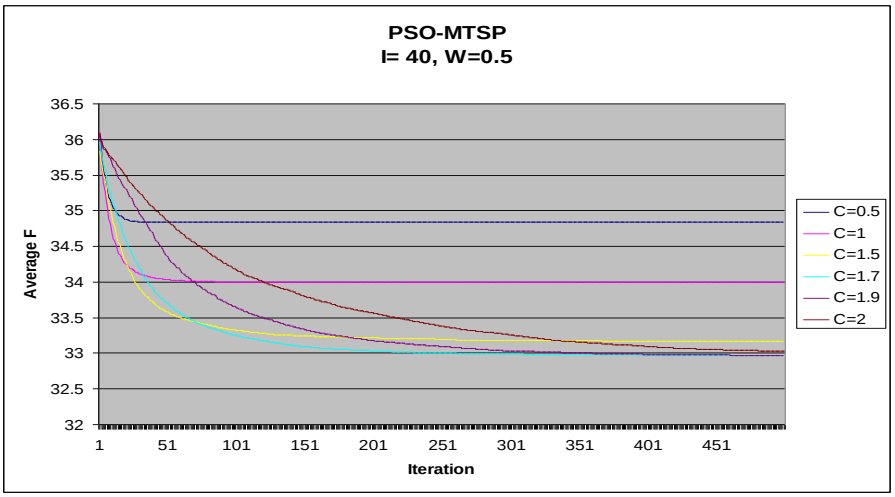
	M_1	M_2	M_3	M_4	M_5
C_1	1.947919	1.248375	0.007237	3	1.259541
C_2	6.259338	6.000145	6.257239	7.000042	6.259552
C_3	12.50724	11.25001	12.50725	11.25957	12.50724



a) swarm size



b) *w* value



c) c_1, c_2 value
Figure (1): Determine the swarm parameters

5- 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 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 [6]:

1. Encoding of the problem in a binary string.
2. Random generation of a population. This one includes a genetic pool representing a group of possible solutions.
3. Reckoning of a fitness value for each subject. It will directly depend on the distance to the optimum.
4. Selection of the subjects that will mate according to their share in the population global fitness.
5. Genomes crossover and mutations.
6. 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.

GA Algorithm for solving MTSP:
The main steps of the algorithm that solves MTSP by PSO as follows:

1- Initial population:
First, the chromosome defines as a set of starting times for the machines in all cycles. So, *S* is the size of the chromosome equal to *n* multiply by *k* (where n number of machine 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* (where number of chromosomes *Q* and number of population *W*) which satisfy the constraint in (P_r). The value of x_{ircp} generated randomly where $h_{ir} \le x_{ircp} \le H_{ir}$.

2- Other generation:
The value of fitness for each chromosome in the previous population had been calculated. The next generation created by selection the best chromosomes which have greatest value in the objective function. *T* is a percent of the previous population which determine the number of best chromosomes that transfer to the next generation.

3- Crossover:
The remaining chromosomes divided as a pair. Each chromosome in each pair divided in certain cycle *v* then swap between *v* to k cycle in first chromosome and *v* to *k* cycles in the second chromosome.

4- Mutation:
The symbol *E* is the number of genes that will be mutated. For each chromosome, the selection of gene which mutated generated randomly. Then the value of these genes will be generated randomly based on the gene boundaries.

The steps form 2 to 4 will be repeated until the last number of population. The solution is the first chromosome of the last population.

GA-MTSP Algorithm:
A1: Reformulate the boundaries for each machine in each cycle as follows:

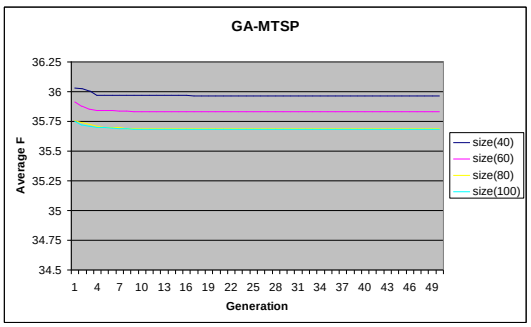
Put $H_{ik} = L_{ik} \quad \forall i \in N$, $h_{jr} = l_{jr} \quad \forall j \in N$,

$H_{jr} = \min(L_{jr}, (\min_{i \in U_j} H_{ir+1} - p_j))$ Where, $U_j = \{i \in N : j \in N^{(i)}\} \quad r = k-1,...,2,1$

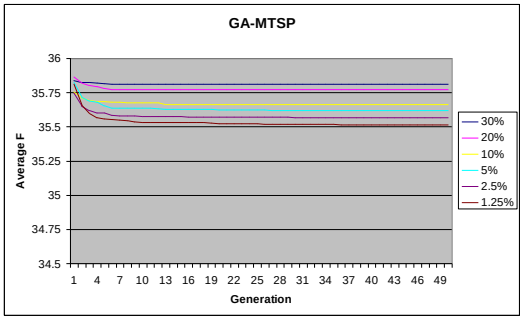
GA2: Put $p = 1$.
GA3: Put $c = 1$.
GA4: Put $r = 1$.
GA5: Put $i = 1$.
GA6: If $r \neq 1$ then $h_{ir+1} = \max_{j \in N^{(i)}} (x_{ircp} + p_{jr}) \quad \forall i \in N$
GA7: Generate random number for x_{ircp} where $h_{ir} \leq x_{ircp} \leq H_{ir}$.
GA8: If $i < n$ then $i = i + 1$ go to GA6.
GA9: 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 $c = Q - (T*Q)$.
GA21: Put $e = 1$.
GA22: Generate random number m between 1 and S.
GA23: Generate random number for x_{mrcp} where $h_{ir} \leq x_{mrcp} \leq H_{ir}$.
GA24: if $e < E$ then $e = e + 1$ go to GA22.
GA25: If $p < W$ then $p = p + 1$ go to GA3.
GA36: The solution is $x_{ir1p} \quad \forall i \in n, \forall r \in k.$

Example:
The previous example solved by GA-MTSP algorithm. The population size has been tested by 40, 60, 80 and 100 chromosomes. The result shows that, the best population size equal 80 chromosome as shown in the figure (2-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, we found that, the best number of chromosomes that will be kept in the next population equal to 1.25% from population size as shown in figure (2-b). When we test the value of crossover probability by 20%, 33% and 50% from the chromosome size, we found that the best value of crossover probability is 33% of the chromosome size as shown in figure (2-c). We found that mutation probability that gives best solution is 10% of chromosome size after try 5%, 10% and 20% of chromosome size. Finally, the Genetic algorithm parameters have been determined that gives the best value for the objective function f (equal 35.32834) and the best starting times is:

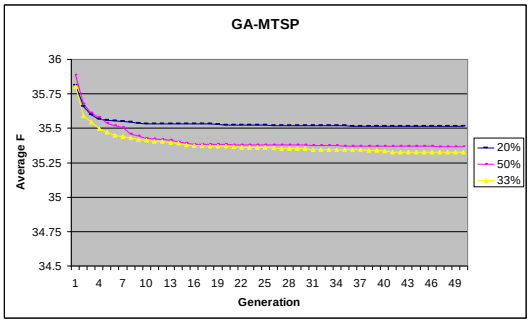
	M_1	M_2	M_3	M_4	M_5
C_1	1.175617	0.861839	0.068779	3.108429	1.357962
C_2	6.487444	6.065244	6.674041	7.203728	6.48237
C_3	12.99107	11.268875	13.112708	11.497321	13.295105



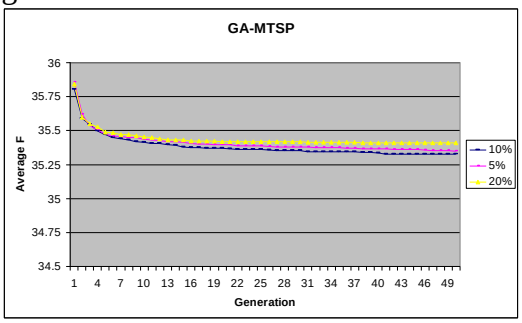
a) Population size



b) No. of chromosomes keeps in next generation



c) Crossover probability



d) Mutation probability

Figure (2): Determine the Genetic algorithm parameters

From previous experimental results we found that, when MTSP solved by particle swarm optimization algorithm, the best value of *f* equal 32.96286, which reached in iteration 400 that take 11 second. But when MTSP solved by genetic algorithm, the best value of *f* equal 35.32834, which reached in generation 41 that take 2 second. But when MTSP solved by max-separable algorithm, the value of *f* equal to 35.75 that take <0.5 (figure (3)).

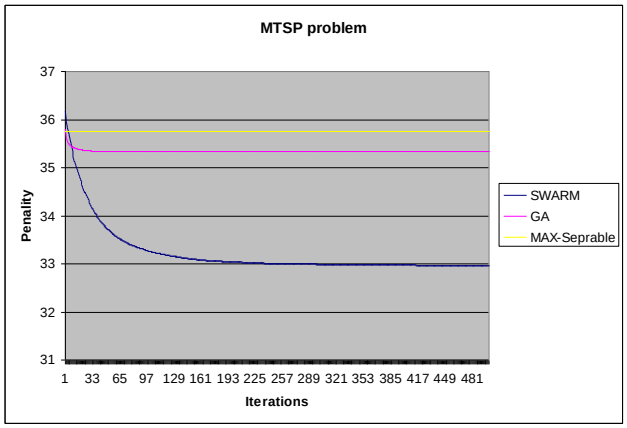


Figure (3): The result of solving MTSP problem by SWARM, GA and Max-separable

6- Conclusion:

The machine time scheduling problem had been solved by particle SWARM optimization, max-separable technique and Genetic algorithm. We founded that, particle SWARM optimization gives best starting time for each machine in each cycle, which gives the lower penalty of the MTSP problem that take more time than other two algorithms. But the Genetic algorithm and max-separable algorithm give high penalty value of the MTSP problem than SWARM algorithm but these two algorithms take low time than SWARM algorithm. That mean, the SWARM gives best objective function than GA and Max-separable but its take

more time than other two algorithms. The cost deference between the cost of objective function and time cost leads the decision maker chooses the best algorithm in this study.

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- 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

5- 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

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

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

8- 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"

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

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

12- 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.