Hybrid Particle SWARM Optimization for Solving Machine Time Scheduling Problem

A. A. El-sawy
Computer Science Dept.,
Faculty of Computers and Information, Banha University,
Elqalubia, Egypt.
E-mail: ahmed_el_sawy {at} yahoo.com

A. A. Tharwat
Decision Support Dept.,
Faculty of Computers and Information, Cairo University,
Cairo, Egypt,

***Abstract*— A hybrid particle swarm optimization (PSO) for multi-machine time scheduling problem (MTSP) with multi-cycles is proposed in this paper to choose the best starting time for each machine in each cycle under pre-described time window and a set of precedence machines for each machine; to minimize the total penalty cost. We developed hybrid algorithm by using a combination between PSO and Genetic Algorithms (GA), precisely the GA operators' crossover and mutation. Based on experimental results for the developed hybrid algorithms, we can conclude that, the algorithm that combines PSO with mutation gives best solution for MTSP.**

Keywords- Machine Time Scheduling, Particle SWARM optimization, Genetic Algorithm, Mutation, Crossover, Time Window

I. INTRODUCTION

Machine Time Scheduling Problem (MTSP was investigated in [20] as a parameterized version of the MTSP, which was defined in [9], with penalized earliness in starting and lateness in the completion of the operation. The authors in [20] applied the optimal choice concept which is given in [17] and some theoretical results from [18] to obtain the optimal values of the given parameters.

In [4] 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 [3] an algorithm was developed to solve a 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 considered deterministic. Authors in [15] discussed how to solve the MTSP when the processing time for each machine is stochastic, and they suggest the Monte Carlo simulation technique to solve the problem. A generalization was introduced in [2] to overstep the cases at which an empty feasible set of solutions is described by the system.

In [1] introduced an algorithm by using the PSO and GA to solve MTSP, and compressions were made between PSO, GA and max-separable technique (using numerical example). The Authors found that PSO algorithm reach to the best solution faster than both the GA and the max-separable technique. In this paper we will introduce a three hybrid algorithms. The first one combines PSO with both the crossover and the mutation operators (HPSOCM-MTSP), while the second algorithm combines PSO with only the crossover operator (HPSOC-MTSP), and the third combines PSO with the mutation operator only (HPSOM-MTSP). The introduced experimental results show that the third combined algorithm HPSOM-MTSP gives the best result among the other algorithms and also superior of the algorithm that developed in [1]. In other words, the HPSOM-MTSP found the best starting time for each machine in each cycle under the pre-described time windows the set of precedence machines to minimize the total penalty.

II. PROBLEM FORMULATION

In multi-machine time scheduling problem with multi-cycles, there are n machines, each machine carries out one operation j with deterministic processing time p_j for $j \in N = \{1,...,n\}$, the machines work in k cycles, and the processing time for each machine does not depend on the cycle number. Let x_{jr} represents the starting time of the j^{th} machine in the rth cycle $\forall \quad j \in N$, $r \in K = \{1,...,k\}$ (k yhe cycle's number). Machine j can start its work in cycle r only after a predecessors set of machines $N^{(j)}, N^{(j)} \subset N$ 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:

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

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

$$\begin{aligned} &\max_{j \in N^{(i)}} (x_{jr} + p_{jr}) \leq x_{ir+1} \quad \forall i \in N, \\ &l_{jr} \leq x_{jr} \leq L_{jr} \qquad \forall j \in N \end{aligned} \qquad (2)$$

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

$$[x_{jr}, x_{jr} + p_j] \subset [a_{jr}, b_{jr}], \quad (3)$$

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

$$f(x) = \max_{j \in N} f_{jr}(x_{jr}) \rightarrow \min \qquad r \in K \qquad (4)$$

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

$$f_{jr}(x_{jr}) = \max\{ f_{jr}^{(1)}(x_{jr}), f_{jr}^{(2)}(x_{jr} + p_{jr}), 0\} \quad \forall j \in N \quad (5)$$

Where $f_{jr}^{(1)} : R \rightarrow R$ 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 $f_{jr}^{(2)}(b_{jr}) = 0$

Based on the above discussion, the main objective of the proposed problem is

"The minimization of the maximum total penalty among all cycles"

That leads to solve the following optimization problem:

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

subject to :

$$\max_{j \in N^{(i)}} (x_{jr} + p_{jr}) \le x_{ir+1} \quad \forall i \in N \qquad (6)$$

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

III. CROSSOVER AND MUTATION:

GA maintains a set of candidate solutions called population and repeatedly modifies them. In each iteration of the GA algorithm, some individuals are selected from the current population to parent and use them for producing 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 [15]. Two operators in GA have been used to generate next generation called crossover and mutation.

In crossover**,** pairs of strings are picked at random from the population to be subjected to crossover. The simplest approach is the single-point crossover that assuming that L is the string length, it randomly chooses a crossover point that takes values in the range 1 to *L* - 1.

The portions of the two strings beyond this crossover point are exchanged to form two new strings [8]. Another approach for the crossover is the two-point crossover, it is usually used if the crossover points are *i* and *j* all the parameter pairs between *i* and *j* have to do the crossover and the new solutions has been generated. The third approach for crossover is the uniform crossover, first a random crossover point is generated and the crossover process is done from this selected point to either the end point or the start point of the solution, which is randomly selected [19].

By mutation individuals are randomly altered. These variations (mutation steps) are mostly small. They will be applied to the variables of the individuals with a low probability (mutation probability or mutation rate). Normally, offspring are mutated after being created by recombination.

For the definition of the mutation steps and the mutation rate two approaches exist: in the first one, both parameters are constant during a whole evolutionary run. In the second, one or both parameters are adapted according to previous mutations.

In mutation, a bit involves during flipping it: changing a 0 to 1 or vice versa. The parameter Pm (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 occurred [8].

IV. 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 [16].

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:

$$v_{j,g}^{(t+1)} = w v_{j,g}^{(t)} + c_1 r_1 (pbest^{(t)}{}_{j,g} - x_{j,g}^{(t)}) + c_2 r_2 (gbest^{(t)}{}_g - x_{j,g}^{(t)}) \qquad (7)$$

$$x_{j,g}^{(t+1)} = x_{j,g}^{(t)} + v_{j,g}^{(t+1)} \qquad (8)$$

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 along the g[th] dimension 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 particle j along the g[th] dimension at iteration t;

$pbest^{(t)}_{j,g}$ = is the best previous position along the g[th] dimension of particle j in iteration t

$gbest^{(t)}_g$ = is the best previous position among all the particles along the g^{th} dimension in iteration t

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 [16].

[13] Describes evolutionary optimization algorithm that is based on particle swarm but with addition of crossover which is one of the evolution operators in genetic algorithm. A comparison between "PSO" and "PSO that includes Crossover" will be made. PSO is known to find global optima for a given solution; while GA is also a search heuristic that finds solution to a given problem. Experiments were discover that for some problems "Standard PSO" works better while in other problem "PSO and Crossover" improve PSO in finding global optima.

[7] 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.

A novel approach based on Particle Swarm Optimization (PSO) for scheduling jobs on computational grids introduced in [6]. 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.

[11] 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.

Mutation operation was combined with basic particle swarm optimization (BPSO) in [5] let's called PSOM. and in [12] and [10] combined crossover with BPSO, let's called PSOC. In [14] used mutation and crossover with BPSO to make a new algorithm, let's called PSOMC. The authors in [5], [12], [10] and [14] proved that their new algorithms are better than BPSO algorithm by tested benchmark functions. The common benchmark functions between these papers as following:

1- Sphere function

$$\text{Min } f_1(x) = \sum_{i=1}^{n} x_i^2 \quad -5.21 \le x_i \le 5.21$$

2- Rosenbrock function

$$\text{Min } f_2(x) = \sum_{i=1}^{n} 100(x_{i+1} - x_i^2)^2 + (x_i + 1)^2 \quad -30 \le x_i \le 30 \tag{9}$$

3- Ackley's function

$$\text{Min } f_3(x) = 20 + e - 20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i-1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) \quad -32 \le x_i \le 32 \tag{10}$$

Table (1) represents the mean fitness for each function in each algorithm. We found that PSOC gives best value for f1 and PSOCM gives best value for f2 and PSOM gives best value for f3.

TABLE I. THE OBTAINED RESULTS FOR PSOM, PSOC AND PSOCM FOR F_1, F_2 AND F_3

	Mean fitness		
	PSOM	***PSOC***	***PSOMC***
f_1	3.327 E-9	**8.517991 E-43**	3.4003 E-40
f_2	106.939	8.608759 E-19	**7.15891 E-29**
f_3	**3.152 E-12**	6.292307 E-10	0.00

With a detailed study for the above results; we found that one can't grantee which algorithm gives the best result. So, three hybrid algorithms are develop for solving MTSP. In the first algorithm, we combine PSO with crossover and mutation operators (HPSOCM-MTSP). While in the second algorithm, the combination was made between PSO with crossover operator (HPSOC-MTSP). Finally in the third algorithm, PSO combined with mutation operator (HPSOM-MTSP).

V. HYBRID PSO ALGORITHM FOR SOLVING MTSP

In this section, the combination of a crossover procedure and/or mutation procedure with a PSO algorithm to develop hybrid PSO algorithm

A. Hybrid PSO Algorithm (HPSO)

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 (particle member) 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 the number of iterations) which satisfy the constraints in (P).

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

1- *Reformulation:*

For each machine j, calculate the set of successors U_j. Then calculate the minimum of the upper bounds for the j^{th} machine's successors, and also calculate the difference between the calculated minimum and its processing time. Therefore the new upper bound of the j^{th} machine will be reformulated as the minimum between its old upper bound and the calculated difference.

2- *Initial iteration:*

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- *New PSO iteration:*

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

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

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.

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

5-*Execute a Crossover operator and/or a Mutation operator on the particles.*

6-*Repeat* step 3 to step 5 until reaching the given number of iterations *T*.

Crossover operation:

The swarm divided into pairs, in each pair there are two particles, divide each particle at a certain position *s*, and then swap between the *s* to k members in both particles, where *k* represents the total number of members in a particle.

Mutation operation:

Use the uniform mutation, which means that, each member in the particle has the same chance to be mutated. The symbol *A* represents the number of mutated particles and the symbol *E* represents the number of mutated members in each particle. The obtained value of the mutated members will be generated randomly based on its boundaries.

Three proposed hybrid algorithms and a comparison between them based on the value of objective function is going to introduced.

B. HPSOCM-MTSP algorithm:

This hybrid algorithm combines the crossover and mutation with PSO as follows (see the flowchart in Appendix A):

1- *Reformulation*: machine boundaries.
2- *Initial iteration:* generate the particles randomly based on new machines boundaries.
3- *Next PSO iteration:* modify the velocity and position of particles.
4- *Calculate* the objective function then find the $pbest_p$ and *gbest*.
5- *Crossover* the particles with each others.
6- *Mutation* the particles
7- *Repeat* step 3 to step 6 until the *T*.

HPSOCM-MTSP Algorithm:

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

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

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

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

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: $pbest_p = f(x_{irpt}) \; \exists i = 1,..,N, \exists r = 1,..,K$.
A11: If *p* < *Q* then *p* = *p* + 1 go to A4.
A12: find min($f(pbest_p)$) $\exists p = 1,..,Q.$
A13: $gbest = pbest_{p_{\min}}$
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 $f(x_{irpt}) > f(x_{irpt-1})$ then $pbest_p = x_{irpt}$
A20: If $p < Q$ then $p = p + 1$ go to A16.
A21: if $f(gbest) > f(pbest_{p_{\min}})$ then $gbest = pbest_{p_{\min}}$.
A22: $f(gbest_t) < f(gbest_{t-1})$ then go to 15.
A23: Put p=1.
A24: Put $r = trunc(s/N) + 1$.
A25: Put $i = rem(s/N)$.
A26: Swap between x_{irpt} and $x_{ir(p+1)t}$.
A27: If $i < n$ then $i = i + 1$ go to A26.
A28: If $r < k$ then $r = r + 1$, i =1 go to A26.
A29: If $p < Q$ then $p = p + 2$ go to A24.
A30: Generate random number A between 1 and Q.
A31: a=1.
A32: Put e = 1.
A33: Generate random number E between 1 and D.
A34: Put $r = trunc(m/N) + 1$.
A35: Put $i = rem(m/N)$.
A36: Generate random number for x_{irat} where $h_{ir} \le x_{irat} \le H_{ir}$.
A37: if $e < E$ then $e = e + 1$ go to A33.
A38: if $a < A$ then $a = a + 1$ go to A32.
A39: if $f(gbest) > f(x_{irat})$ then $gbest = x_{irat}$.
A40: If $t < T$ then go to A15.
A41: The solution is *gbest.*

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

VI. NUMERICAL 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 II. UPPER AND LOWER BOUNDARIES FOR THE MACHINE'S STARTING TIME

Cycle (r)	r = 1	r = 2	r = 3
lir $i = 1,2,..,5$	{1,0,0,3,1}	{4,6,6,5,6}	{10,11,12,9,11.5}
Lir $i = 1,2,..,5$	{5,4,3,5,6}	{6.5,7,7.5,7.25,6.5}	{13,12,15,12,14}

TABLE III. MACHINE'S PREDECESSORS AND MACHINE'S SUCCESSORS

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 to represent the end points of the tolerance interval for each machine $[a_{jr}, b_{jr}]$, so that we have in our case for all $j \in N$

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

Input values of a_{ir} and b_{ir} for each cycle are given in table (4).

TABLE IV. THE END POINTS OF THE TOLERANCE INTERVALS FOR THE MACHINE'S STARTING TIME

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 running the program 100 trials and taking the average of objective function's values and call it *averageF* (values calculated for each iteration in each trial), it is found that; the best parameters for the HPSO are: the swarm size equals 80, the value of w equals 0.5 and the values c_1 *and* c_2 equal 1.7. concerning the cutting point in the a particle to crossover, it is found that the best position of cutting point is 33% of the particle size, that means, the cutting should be occurred between cycles. Concerning the mutation, it is found that the mutation probability equals 10% of particle size and also the number of particles which will be mutated should equal 20% of the swarm size.

The proposed hybrid HPSOCM-MTSP algorithm which combines the crossover and mutation procedures with the HPSO has been applied on the illustrative example (scenario I) and it is found that the minimum value of *averageF* which is 33.08 arises after 4400 iterations as shown in the figure 1 (with no improvement of the value whatever the number of iterations increases).

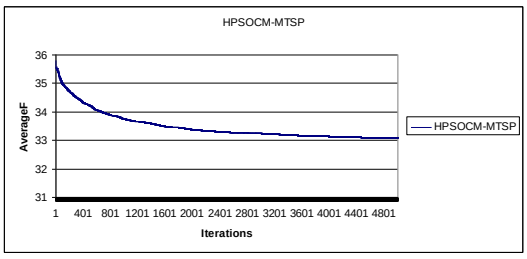


Figure 1. The solution of HPSOCM-MTSP

When HPSOCM-MTSP without the mutation operation has been applied (i.e by applying the crossover operation only) on the same example (scenario II), it is found that the minimum value of *averageF* which is 34.97 arises after 220 iterations as shown in the figure 2 (with no improvement of the value whatever the number of iterations increases).

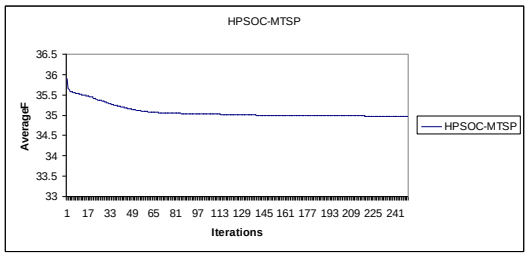


Figure 2. The solution of HPSOC-MTSP

Finally HPSOCM-MTSP without the crossover operation has been applied (i.e by applying the mutation operation only) on the same example (scenario III), it is found that the minimum value of *averageF* which is 32.83 arises after 380 iterations as shown in the figure 3 (with no improvement of the value whatever the number of iterations increases).

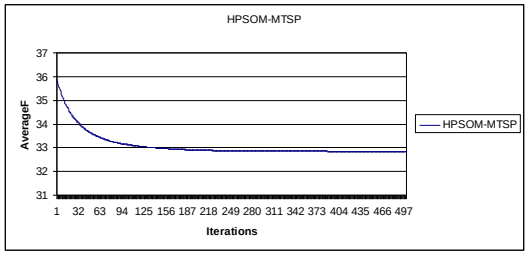


Figure 3. The solution of HPSOM-MTSP

So the best starting times that are given by applying (scenario III) is represented in table (5) with the best value of *averageF* (which is 32.83)

TABLE V. THE BEST STARTING MACHINE'S STARTING TIME FOR 5 MACHINES AND THREE CYCLES

Machine Cycle	**M_1**	**M_2**	**M_3**	**M_4**	**M_5**
C_1	1.74	1.7	0.02	3	1.39
C_2	6.27	6	6.27	7	6.39
C_3	12.52	11.25	12.52	11.39	12.52

Now assume that the cost value is $\$\alpha$, then the penalty cost in our case equals $\$32.83\alpha$, with a difference $\$2.14\alpha$ if the crossover operation only has been applied, and with a difference $\$0.25\alpha$ if both the crossover operation and the mutation operation have been applied.

It is worth to mention that the convergence of HPSOM-MTSP which combines the mutation procedure with PSO algorithm has been examined in [4], the authors conclude that the algorithm speed up the convergence for the hardest test functions.

It is very important to mention also, according this article, one can say that, the proposed hybrid algorithm is considered the best one only for this type of problems (MTSP).

VII. CONCLUSION:

A proposed hybrid algorithm HPSOCM-MTSP was developed, and after applying three different scenarios numerically, it is found that the HPSOCM-MTSP algorithm (with applying the Mutation operation only) gives the best result among the three applied scenarios, for solving the proposed machine time scheduling problem, with penalty cost value $\$32.83\alpha$. Moreover the HPSOCM-MTSP proposed algorithm gives a better result comparing with the results that obtained by applying PSO-MTSP algorithm [1], GA-MTSP algorithm [1] and max-separable technique [3]. Figure 4 represents the comparative study between the penalty function values for the proposed algorithm with its three scenarios and the other three algorithms.

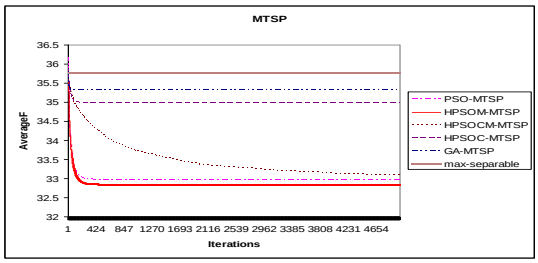


Figure 4. Comparison between HPSOM-MTSP algorithm and other different algorithms

REFERENCES:

[1] A. A. Sawy and A. A. Tharwat, "Comparison of Particle SWARM Optimization, Genetic Algorithm and Max separable Technique for Machine Time Scheduling Problem ", International Journal of Computer Information Systems, Vol. 1, No. 3, pp. 46-52, 2010.

[2] A. Tharwat and A. Abuel-Yazid, "Generalized Algorithm For Muulti-Cycle Machine Time Scheduling", Proceeding of The third Assiut University Int. Conf. On Mech. Eng. Advanced Tech. For Indus. Prod. December 24-26 (Meatip3), Egypt, pp. 602 – 608, 2002.

[3] A. Tharwat and A. Abuel-Yazid: "Multi-Cycles Machine Time Scheduling Problem", International Journal of Intelligent Computing & Information Science (ICIS) vol. 2, No. 1, pp. 78-84, 2002.

[4] A. Tharwat and K. Zimmermann: "Optimal Choice of Parameters in Machine Time Scheduling Problems Case I," Conference MMEI, Liberc, Czech Republic, ISBN 80-7083, 338,6, pp. 107-112, 1998.

[5] Andrew Stacey, Mirjana Jancic, Ian Grundy " Particle Swarm Optimization with Mutation" Proceedings of the IEEE Swarm Intelligence Symposium SIS03 IEEE Press, Vol. 2, pp. 1425-1430, 2003

[6] 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

[7] I. H. Grundy and A. Stacey, "Particle swarm optimization with combined mutation and hill climbing" Proceedings of 7th Asia-Pacific Conference on Complex Systems, Cairns, Australia, vol. 12, pp.1- 10, 2004.

[8] M. Srinivas, Motorola India Electronics Ltd. Lalit M. Patnaik, Indian Institute of Science tlmization and search methods ... 0018-9162/QM "Genetic Algorithms: A Survey", IEEE, pp.17-24, 1994.

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

[10] Millie Pant, Member, IAENG, Radha Thangaraj, Member, IAENG, and V. P. Singh "Particle Swarm Optimization with Crossover Operator and its Engineering Applications" IAENG International Journal of Computer Science, Vol. 36, No. 2, on line publication, 2009

[11] 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.

[12] Millie Pant, Radha Thangaraj and Ajith Abraham "A New PSO Algorithm with Crossover Operator for Global Optimization Problem" INNOVATIONS IN HYBRID INTELLIGENT SYSTEMS Advances in Soft Computing, Vol. 44, pp. 215-222, 2007.

[13] Olapeju Latifat Ayoola "Particle Swarm Optimization and Crossover" COMP40580 Natural Computing- Mini-Project Papers, on line publication, 2006.

[14] Qing Zhang, Changhe Li, Yong Liu and Lishan Kang "Fast Multi-swarm Optimization with Cauchy Mutation and Crossover Operation" ADVANCES IN COMPUTATION AND INTELLIGENCE Lecture Notes in Computer Science, Vol. 46, pp. 344-352, 2007.

[15] 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 , pp. 103 – 110, 10-12 Sept 2003.

[16] S. Panda and N. P. Padhy, "Comparison of Particle Swarm Optimization and Genetic Algorithm for TCSC-based Controller Design", International Journal of Electrical and Electronics Engineering 1:5 2007, Vol. 1, No. 5, pp. 305-313 2007

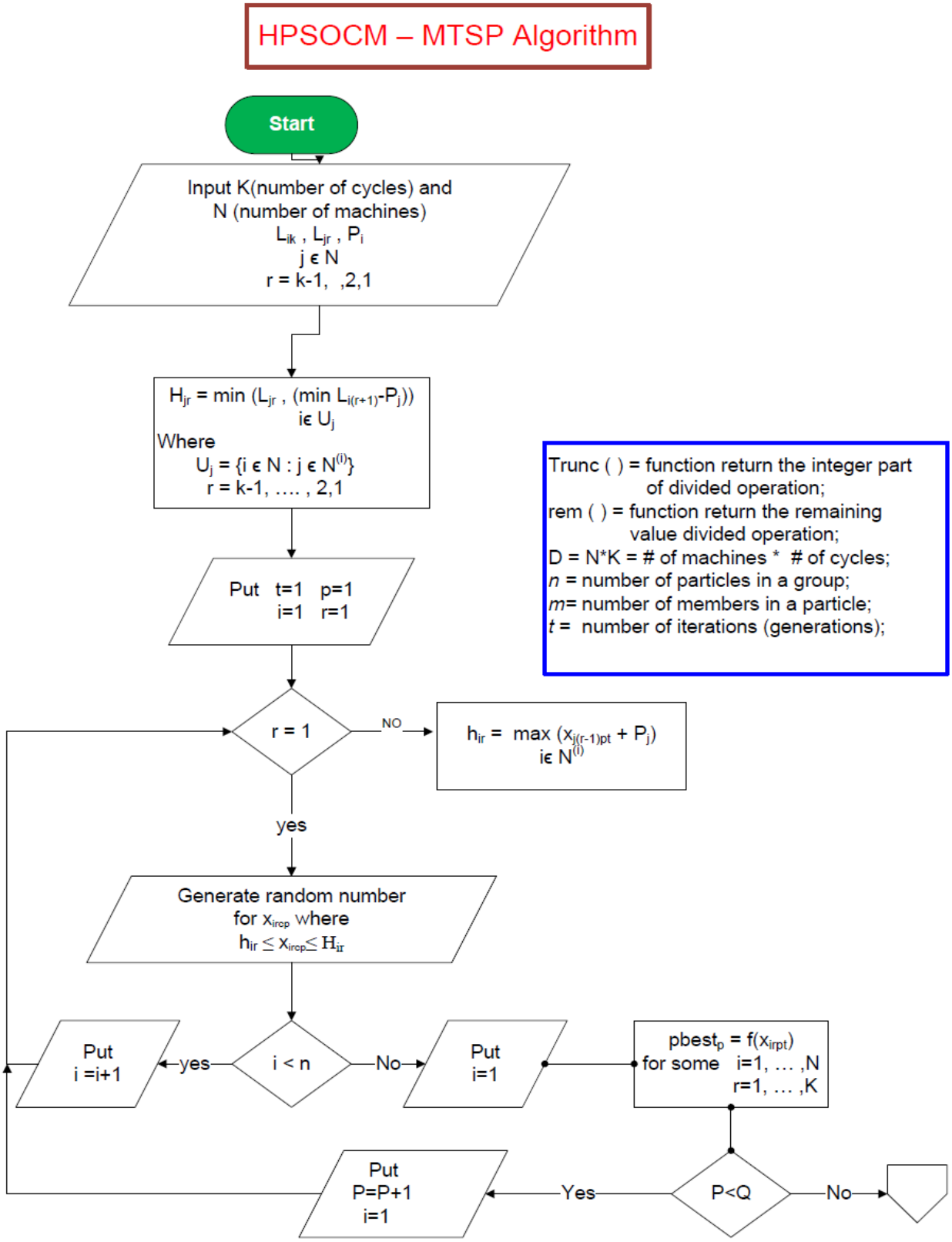
[17] S. Zlobec: "Input Optimization I: Optimal Realizations of Mathematical Models," Mathematical Programming, Vol. 31, pp.245-268, 1985.

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

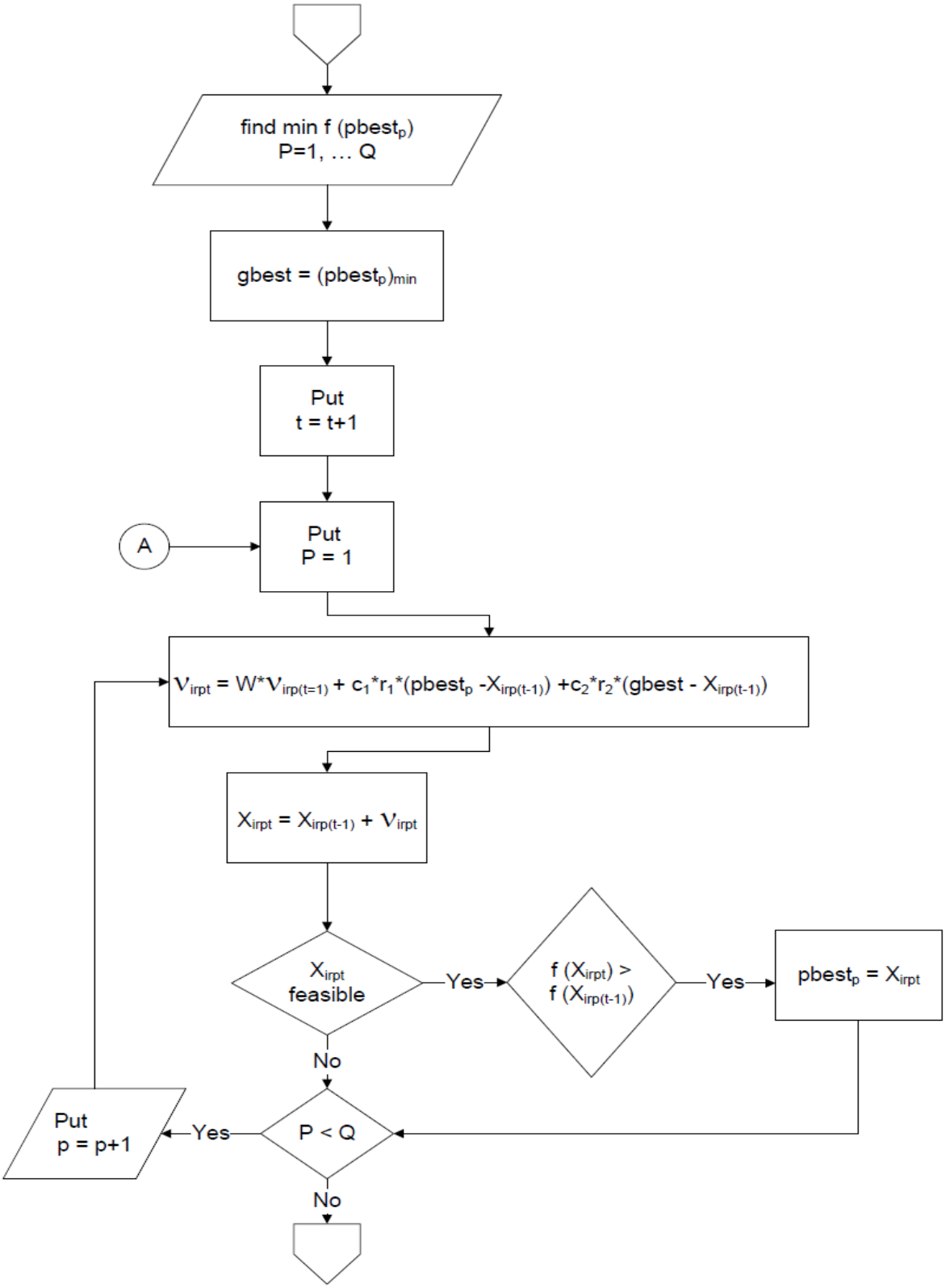
[19] Shun-Fa Hwang, Rong-Song He "Improving real-parameter genetic algorithm with simulated annealing for engineering problems" Advances in Engineering Software, Vol. 37, No. 6, pp. 406-418, 2006.

[20] 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, Vol. 33, No. 1, pp. 53-61, 1992.

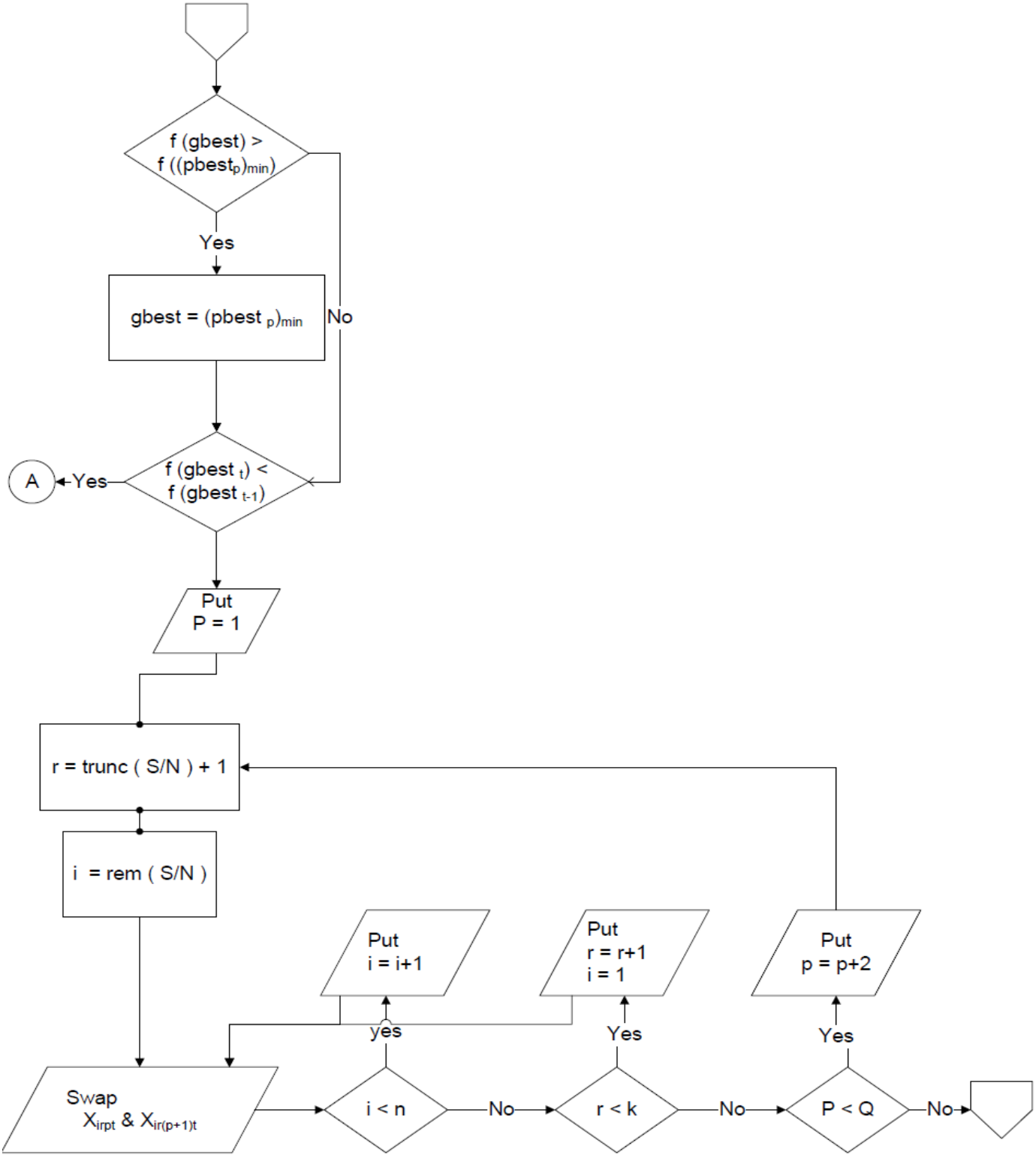
Appendix A



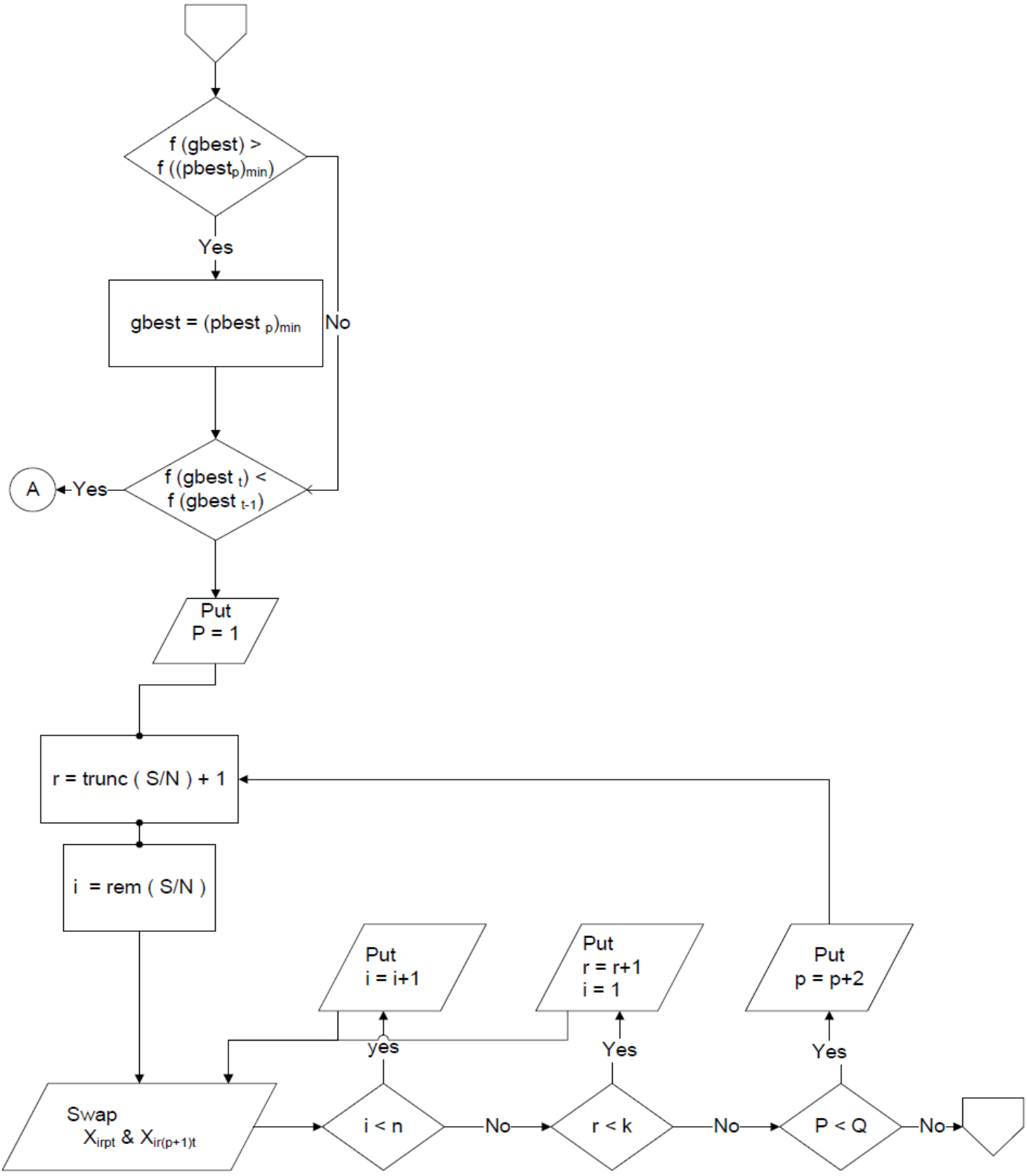
HPSOCM-MTSP Algorithm Flowchart



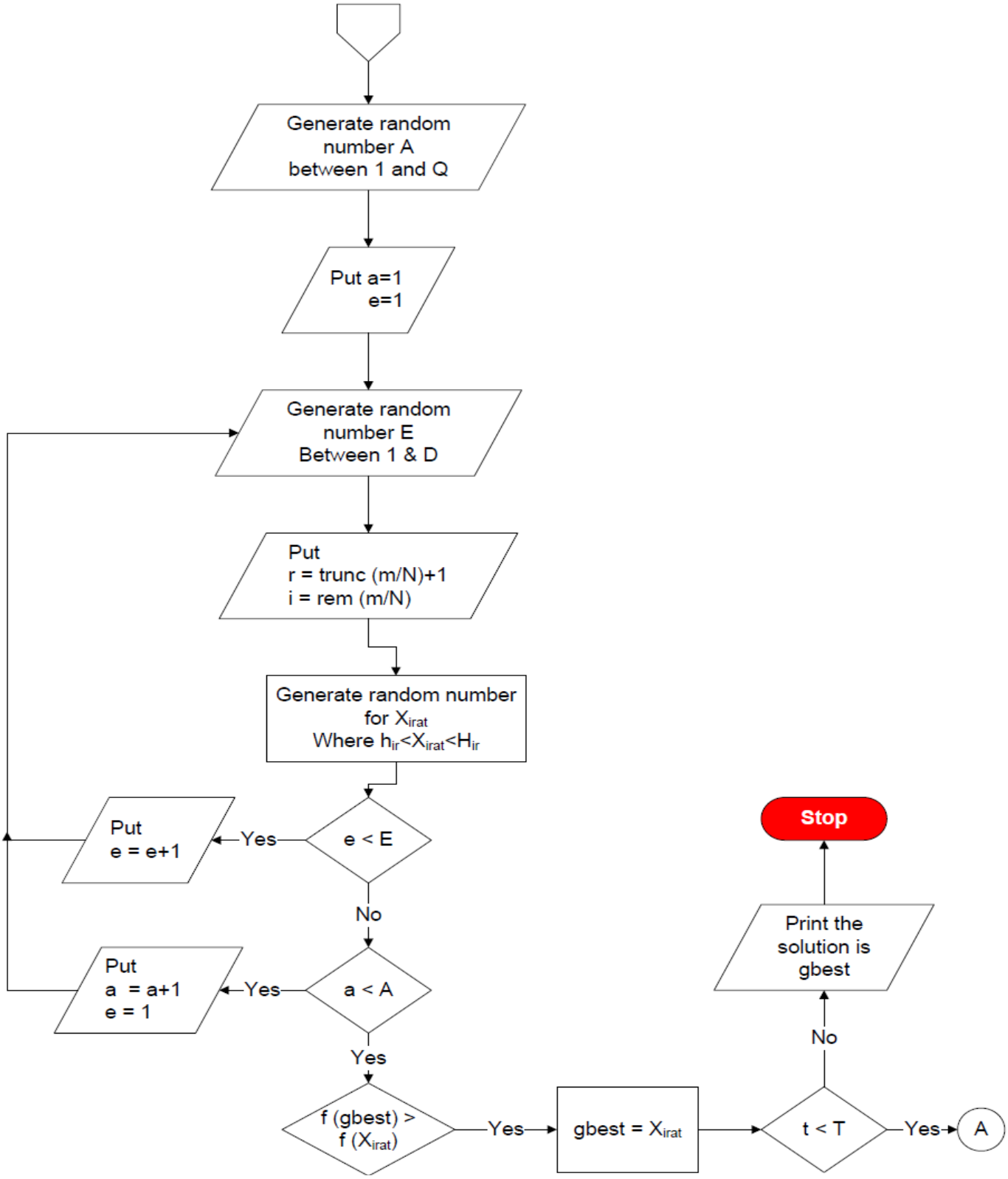
HPSOCM-MTSP Algorithm Flowchart (cont'd)



HPSOCM-MTSP Algorithm Flowchart
(cont'd)



HPSOCM-MTSP Algorithm Flowchart
(cont'd)



HPSOCM-MTSP Algorithm Flowchart (cont'd)