Journal of Industrial and Systems Engineering
Vol. 1, No. 4, pp 304-317
Winter 2008

JISE

A New Solution for the Cyclic Multiple-Part Type Three-Machine Robotic Cell Problem based on the Particle Swarm Meta-heuristic

N. Kamalabadi[1*], S. Gholami[2], A. H. Mirzaei[2]

[1]Department of Industrial Engineering, University of Kurdistan, Sanandaj, Iran, email: nakhai_isa@yahoo.com

[2]Department of Industrial Engineering, Tarbiat Modares University, Tehran, Iran

ABSTRACT

In this paper, we develop a new mathematical model for a cyclic multiple-part type three-machine robotic cell problem. In this robotic cell a robot is used for material handling. The objective is finding a part sequence to minimize the cycle time (i.e.; maximize the throughput) with assumption of known robot movement. The developed model is based on Petri nets and provides a new method to calculate cycle times by considering waiting times. It is proved that scheduling problem of a robotic cell is unary NP-complete. Achieving an optimal solution for this type of complex, large-sized problem in reasonable computational time by using traditional approaches and optimization tools is extremely difficult. In this paper we implement an algorithm based on the particle swarm optimisation (PSO) method for solving the problem. To validate the developed model and solution algorithm, various test problems are examined some of which are of small-size and some other of large-size. The computational results show that the proposed algorithm achieves optimum solutions for small sized problems, while for large-sized problems this algorithm can find suitable solutions in acceptable time.

Keywords: Cyclic blocking flow-shop, Particle swarm optimisation, Robotic cell, Scheduling

1. INTRODUCTION

Nowadays the level of automation in manufacturing industries has increased dramatically. Some examples of such progress in automation can be seen in cellular manufacturing and robotic cells. A growing body of evidence suggests that, in a wide variety of industrial settings, material handling within a cell can be accomplished very efficiently by employing robots (Asfahl, 1992). Among the interrelated issues to be considered in using robotic cells are their designs, the scheduling of robot moves, and the sequencing of parts to be produced.

Robotic cell problems in which robots are used as means of material handling have received considerable attention. Sethi et al. (1992) proved that in buffer-less single-gripper two-machine robotic cells producing single part-type and having identical robot travel times between adjacent machines and identical load/unload times, a 1-unit cycle provides the minimum per unit cycle time in the class of all solutions, cyclic or otherwise. For the three-machine case, Crama and van de Klundert, (1999), and Brauner and Finke (1999) have shown that the best 1-unit cycle is the optimal

* Corresponding Author

solution for the class of all cyclic solutions. Hall et al. (1997, 1998) considered the computational complexity of the multiple-type parts three-machine robotic cell problem under various robot movement policies. This problem is studied for no-wait robotic cells too. For example Agnetis (2000) found an optimal part schedule for no-wait robotic cells with three and two machines. Agnetis and Pacciarelli (2000) have studied part-scheduling problem for no-wait robotic cells, and found the complexity of the problem. Crama et al. (2000) studied flow-shop scheduling problems, including the models and their complexity. Dawande et al. (2005) reviewed recent developments in robotic cells and, provided a classification scheme for robotic cell scheduling problems. What follows include some other special cases which have been studied by others: Drobouchevitch et al. (2006) provided a model for cyclic production in a dual-gripper robotic cell. Gultekin et al. (2006) studied robotic cell scheduling problem with tooling constraints for a two-machine robotic cell where some operations can only be processed on the first machine and some others can only be processed on the second machine and the remaining can be processed on both machines. Gultekin et al. (2007) considered a flexible manufacturing robotic cell with identical parts in which machines are able to perform different operations and the operation time is assumed a variable and not a system parameter. They proposed a lower bound for 1-unit cycles and 2-unit cycles. Sriskandarajah et al. (1998) classified the part sequence problems associated with different robot movement policies. In this paper a robot movement policy is considered, whose part scheduling problem is NP-Hard. This is the problem that Baghchi et al. (2006) proposed to solve by a heuristic or meta-heuristic. In this paper a meta-heuristic method based on particle swarm optimization is applied to solve the problem.

In this paper an *m*-machine flexible cyclic cell is considered. All parts in an MPS (A minimal part set) visit each machine in the same order, the production environment is cyclic, and parts are produced in the same order repeatedly.

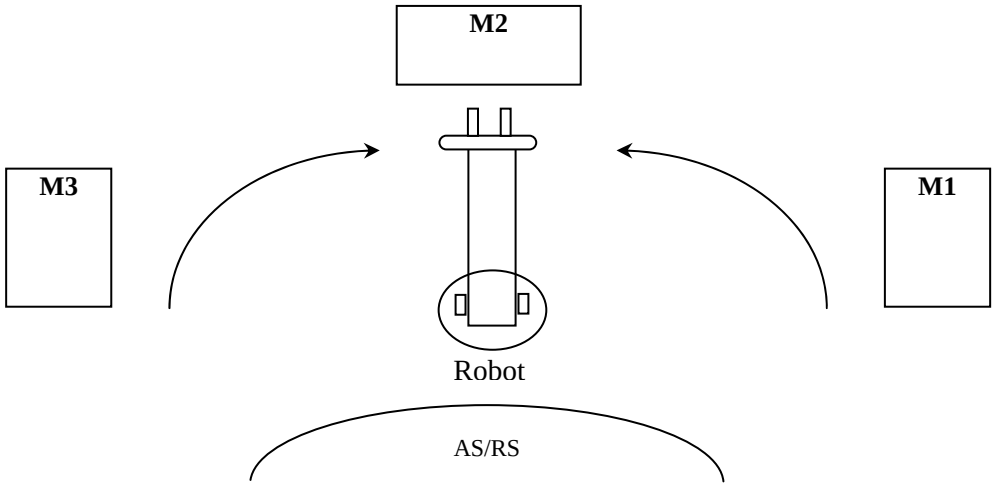

Figure 1. Robotic work cell layout with three machines

In this paper we consider multiple-type part, three-machine robotic cells which have operational flexibility in which the operations can be performed in any order; moreover each machine can be configured to perform any operation. To explain the problem, consider a machining centre where three machine tools are located and a robot is used to feed the machines designated as M_1, M_2, M_3 (see Figure 1). All parts are brought to and removed from the robotic cell by

Automated Storage & Retrieval System (AS/RS). The pallets and feeders of the AS/RS system allow hundreds of parts to be loaded into the cell without human intervention. Machines can be configured to perform any operation.

The aim of this paper is to find a schedule for the robot movement and the sequence of parts to maximize throughput (i.e., to minimize cycle time). As it is shown, this problem is NP-Complete in general (see Hall et al. 1998). Thus a meta-heuristic algorithm, based on particle swarm optimization is proposed for solving this problem. The rest of this paper is organized as follows: The problem definition and required notations are presented in Section 2, Section 3 presents the developed mathematical model, and in Section 4 the implementation of particle swarm optimization algorithm is described. The computational results are reported in Section 5, and the conclusions are presented in Section 6.

2. PROBLEM DEFINITION

The robotic cell problem is a special case of the cyclic blocking flow-shop, where the jobs might block either the machine or the robot. In a cyclic schedule the same sequences repeat over and over and the state of the cell at the beginning of each cycle is similar to the next cycle. It is assumed that the discipline for the movements of parts is an ordinary flow-shop discipline in which a part meets machines M_1, M_2, M_3 consequently.

2.1.Notations

The following notation is used to describe the robotic cell problem:

m	*The number of machines*
I/O	*The automated input-output system for the cell*
PT_i	*The part-type i to be produced*
r_i	*The minimal ratio of part i to be produced*
MPS	*A minimal part set consisting of r_i parts of type PT_i*
n	*The total number of parts to be produced in the MPS ($n = r_1 + r_2 + ... + r_k$)*
a_i	*The processing time of part i on M_1*
b_i	*The processing time of part i on M_2*
c_i	*The processing time of part i on M_3*
δ	*Robot travelling time between two successive machines (I/O is assumed as machine M_0)*
ε	*The load/unload time of part i*
w_i^j	*The robot waiting time on M_i to unload part i*
S^k	*The robot movement policy S under category k*
T^k	*The cycle time under S^k*

In this study the standard classification scheme for scheduling problems: $\psi_1 | \psi_2 | \psi_3$ is used where ψ_1 indicates the scheduling environment, ψ_2 describes the job characteristics and ψ_3 defines the objective function (Dawande et al, 2005). For example $\mathrm{FRC}_3 | \mathrm{k} \geq 2, \mathrm{S}^1 | \mathrm{C}_\mathrm{t}$ denotes the

minimization of cycle time for multi-type part problem in a three flow-shop robotic cell, restricted to robot move cycle S^1.

2.2.Three Machine Robotic Flow Shop Cell $FRC_3 \mid k \geq 2 \mid C_t$

In the three machine robotic flow shop cell, there are six different potentially optimal policies for the robot to move the parts between machines (Bagchi et al, 2006). Sethi et al. (1992) showed that any potentially optimal one-unit robot move cycle in a m machine robotic cell can be described by exactly *m*+1 following basic activities:

M_i^- Load a part on M_i, $\quad i = 1,2,...,m$

M_i^+ Unload a finished part from M_i, $\quad i = 1,2,...,m$

In other words, a cycle can be uniquely described by a permutation of *m*+1 activities. The following are the available robot move cycles for m=3 flow-shop robotic cell (Sethi et al, 1992):

$$S^1 : \{M^+_3, M^-_1, M^-_2, M^-_3, M^+_3\}$$
$$S^2 : \{M^+_3, M^-_1, M^-_3, M^-_2, M^+_3\}$$
$$S^3 : \{M^+_3, M^-_3, M^-_1, M^-_2, M^+_3\}$$
$$S^4 : \{M^+_3, M^-_2, M^-_3, M^-_1, M^+_3\}$$
$$S^5 : \{M^+_3, M^-_2, M^-_1, M^-_3, M^+_3\}$$
$$S^6 : \{M^+_3, M^-_3, M^-_2, M^-_1, M^+_3\}$$

In this paper we consider a three machine robotic cell problem under the S^6 policy (Figure 2). The problem of finding the best part sequence using the robot move cycle s^6 is NP-complete (Hall et al, 1998).

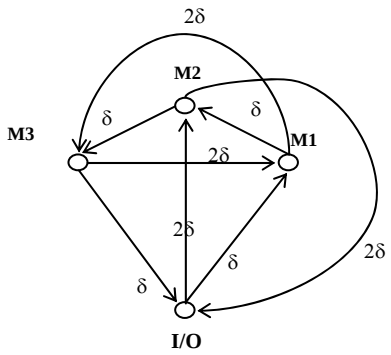

Figure 2. Robot movements under s^6 policy

The following lemma helps to achieve this purpose.

Lemma 1*:* The cycle times of one unit for the policy s^6 are given by:

$$T^6_{\sigma(i)\sigma(i+1)\sigma(i+2)} = 12\delta + 8\varepsilon + \max\{0, a_{\sigma(i+2)} - 8\delta - 4\varepsilon, b_{\sigma(i+1)} - 8\delta - 4\varepsilon, c_{\sigma(i)} - 8\delta - 4\varepsilon\}$$

Proof: According to Figure 2 the robot movement under policy s^6 is as follow:

Pickup part P_{i+2} from $\mathrm{I/O}(\varepsilon)$, move it to $\mathrm{M}_1(\delta)$, load P_{i+2} onto $\mathrm{M}_1(\varepsilon)$, go to $\mathrm{M}_3(2\delta)$, if necessary wait at $\mathrm{M}_3(\mathrm{w}^{i}_{3})$, unload P_i from $\mathrm{M}_3(\varepsilon)$ move it to $\mathrm{I/O}(\delta)$ drop P_i at $\mathrm{I/O}(\varepsilon)$ go to $\mathrm{M}_2(2\delta)$ if necessary wait at $\mathrm{M}_2(\mathrm{w}^{i+1}{}_2)$, unload P_{i+1} from $\mathrm{M}_2(\varepsilon)$, move it to $\mathrm{M}_3(\delta)$, load P_{i+1} onto $\mathrm{M}_3(\varepsilon)$, go to $\mathrm{M}_1(2\delta)$, if necessary wait at $\mathrm{M}_1(\mathrm{w}^{i+2}{}_1)$, unload P_{i+2} from $\mathrm{M}_1(\varepsilon)$, move it to $\mathrm{M}_2(\delta)$, load P_{i+2} onto $\mathrm{M}_2(\varepsilon)$, go to $\mathrm{I/O}(2\delta)$, then start a new cycle by picking up part P_{i+3}.

The cycle time by considering waiting times is as follow:

$$\mathrm{T}^{6}_{\mathrm{I},\sigma(\mathrm{i})\sigma(\mathrm{i+1})\sigma(\mathrm{i+2})} = 12\delta + 8\varepsilon + w_1^{i+2} + w_2^{i+1} + w_3^{i}$$

$$w_1^{i+2} = \max\{0, a_{\sigma(i+2)} - w_2^{i+1} - w_3^{i} - 8\delta - 4\varepsilon\}$$

$$w_2^{i+1} = \max\{0, b_{\sigma(i+1)} - w_3^{i} - 8\delta - 4\varepsilon\}$$

$$w_3^{i} = \max\{0, c_{\sigma(i)} - w_1^{i+2} - 8\delta - 4\varepsilon\}$$

$$\mathrm{T}^{6}_{\sigma(\mathrm{i})\sigma(\mathrm{i+1})\sigma(\mathrm{i+2})} = 12\delta + 8\varepsilon + \max\{0, \mathrm{a}_{\sigma(\mathrm{i+2})}\text{-}8\delta\text{-}4\varepsilon, \mathrm{b}_{\sigma(\mathrm{i+1})}\text{-}8\delta\text{-}4\varepsilon, \mathrm{c}_{\sigma(\mathrm{i})}\text{-}8\delta\text{-}4\varepsilon$$

3. DEVELOPING THE MATHEMATICAL MODEL

In this section we develop a systematic method to produce necessary mathematical programming formulation for the robotic cell problem. First, we model a single-part type problem through Petri nets, and then extend the model to a multiple-part type problem.

A Petri-net is a four-tuple $PN(P, T, A, W)$ where $P = \{p_1, p_2, ..., p_n\}$ is a finite set of places, $T = \{t_1, t_2, ..., t_m\}$ is a finite set of transitions, $A \subseteq (P \times T) \bigcup (T \times P)$ is a finite set of arcs, and $W : A \to \{1,2,3,...\}$ is a weight function. Every place has an initial marking $M_0 : P \to \{0,1,2,...\}$. If time is assigned to transitions then *PN* is called a Petri net.

The behavior of a system can often be described by states of the system and their changes in order to simulate its dynamic behaviour. Marking in a Petri-net is changed according to the following transition (firing) rule:

1) A transition is said to be enabled if each input place p of t is marked at least with $w(p,t)$ tokens, where $w(p,t)$ is the weight of the arc from p to t.

2) An enabled transition may or may not be fired (depending on whether or not the event takes place).

A firing of an enabled transition t removes $w(p,t)$ tokens from each input place p of t and adds $w(t,p)$ tokens to each output place p of t, where $w(t,p)$ is the weight of the arc from t to p.

By considering a single-part type system, the robot arm at the steady state is located at machine M_2, therefore by coming back to this node we have a complete cycle for the robot arm.

The related Petri net for robot movements is shown in Figure 3 and the description of the nodes for this graph with respective execution times would be as follows:

R_1 : Go to $M_3(\delta)$ R_2 : Load $M_3(\varepsilon)$ R_3 : Go to $M_1(2\delta)$

R_4 :Unload $M_1(\varepsilon)$ R_5 : Go to $M_2(2\delta)$ R_6 : Load $M_2(\varepsilon)$

R_7 : Go to input, pickup a new part, go to $M_1(\varepsilon+3\delta)$

R_8 : Load $M_1(\varepsilon)$ R_9 : Go to $M_3(2\delta)$ R_{10} : Unload $M_3(\varepsilon)$

R_{11} : Go to output, drop the part and go to $M_3(\varepsilon+3\delta)$

R_{12} : Unload $M_2(\varepsilon)$

RP_j : Wait at $M_j(w_j^i)$

s_i : starting time of R_i

sp_j : starting time of RP_j

α : M_1 is ready to be unloaded;

β : M_2 is ready to be unloaded;

γ : M_3 is ready to be unloaded;

By considering a multiple-part type system, at machine M_1, when we want to load a part on this machine we have to decide which part should be chosen such that the cycle time is minimized. The same thing also can be achieved for M_2 and M_3. Based on the chosen gate definition we simply have three gates α , β , and γ to choose. Thus we can formulate the following 0-1 integer program:

$$\alpha_1 : s_{4,1} - s_{8,n} + C_t \geq \sum_{i=1}^{n} x1_{in}(a_i) + \varepsilon$$

$$\alpha_j : s_{4,j+1} - s_{8,j} \geq \sum_{i=1}^{n} x1_{ij}(a_i) + \varepsilon \quad j = 2, \cdots, n$$

$$\beta_j : s_{12,j} - s_{6,j} \geq \sum_{i=1}^{n} x2_{ij}(b_i) + \varepsilon \quad j = 1, \cdots, n$$

$$\gamma_j : s_{10,j} - s_{2,j} \geq \sum_{i=1}^{n} x3_{ij}(c_i) + \varepsilon \quad j = 1, \cdots, n.$$

Where $x1_{ij}$, $x2_{ij}$, and $x3_{ij}$, are integer variables.

In order to validate this model the following definition and theorem are presented

Definition. A marked graph is a Petri-net such that at every node (place) it has just one input and one output.

Theorem 1. For a marked graph which has m_i tokens at every node (see Figure 4), the elation $S_B \geq S_A + m_i C_t$ holds, where S_A , S_B are starting times of transitions A and B respectively, and C_t is the cycle time, is true.

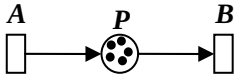

Figure 4. The marked graph in theorem 2

Proof: See Maggot, (1984).

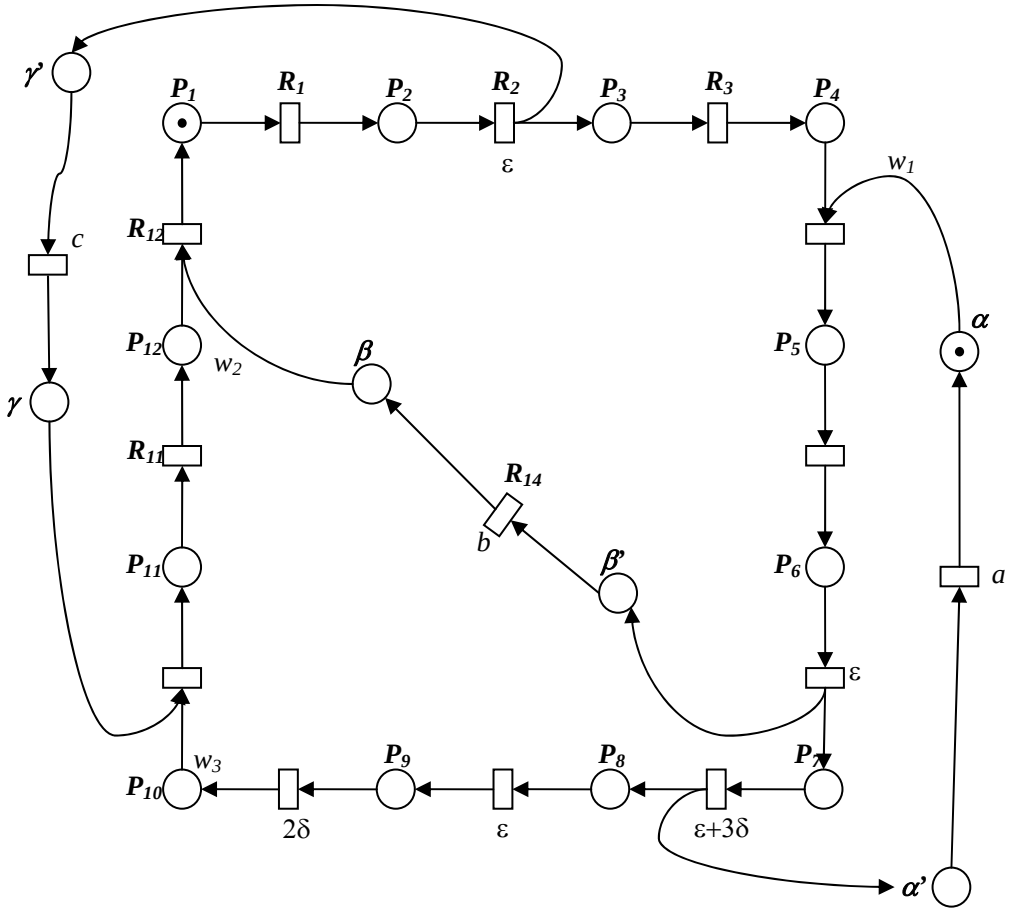

Figure 3. Petri net for S^6 policy

In addition, the following feasibility constraints assign a unique positioning for every job:

$$\sum_{i=1}^{n} x1_{ij} = 1 \quad j = 1,\cdots,n$$

$$\sum_{j=1}^{n} x1_{ij} = 1 \quad i = 1,\cdots,n$$

To keep the sequence of the parts between machines in a right order, we have to add the following constraints:

$$x1_{i,j} = x2_{i,j+1} \quad i = 1,...,n, j = 1,\cdots,n$$

$$x2_{i,j} = x3_{i,j+1} \quad i = 1,...,n, j = 1,\cdots,n$$

Where, we assume that $x1_{i,n+1} = x1_{i,1}$ because of the cyclic repetition of parts.

Thus the complete model for the three machine robotic cell problem with multiple-parts would be as follows:

Min C^6_t

Subject to

$$p_{1,1} : s_{2,1} - s_{12,n} + C_t = \varepsilon + \delta \quad (1)$$

$$p_{1,j} : s_{2j} - s_{12j} = \varepsilon + \delta \quad (2)$$

$$p_{3,j} : s_{4j} - s_{2j} - w_{1j} = \varepsilon + 2\delta \quad (3)$$

$$p_{5,j} : s_{6j} - s_{4j} = \varepsilon + \delta \quad (4)$$

$$p_{7,j} : s_{8j} - s_{6j} = 2\varepsilon + 3\delta \quad (5)$$

$$p_{9,j} : s_{10j} - s_{8j} - w_{3j} = \varepsilon + 2\delta \quad (6)$$

$$p_{11,j} : s_{12j} - s_{10j} - w_{2j} = 2\varepsilon + 3\delta \quad (7)$$

$$\alpha_1 : s_{4,1} - s_{7,1} + C_t - \sum_{i=1}^{n} x1_{in}(a_i) \geq \varepsilon \quad (8)$$

$$\alpha_j : s_{4,j} - s_{7,j} - \sum_{i=1}^{n} x1_{ij}(a_i) \geq \varepsilon \quad (9)$$

$$\beta_j : s_{12,j} - s_{6,j} - \sum_{i=1}^{n} x2_{ij}(b_i) \geq \varepsilon \quad (10)$$

$$\gamma_j : s_{10,j} - s_{2,j} - \sum_{i=1}^{n} x3_{ij}(c_i) \geq \varepsilon \quad (11)$$

$$x1_{i,j-1} = x2_{i,j} \quad (12)$$

$$x2_{i,j-1} = x3_{i,j} \quad (13)$$

$$\sum_{i=1}^{n} x1_{ij} = 1 \quad (14)$$

$$\sum_{j=1}^{n} x1_{ij} = 1 \quad (15)$$

$$s_{ij} \geq 0, w_{kj} \geq 0, \; x1, x2, x3 \in \{0,1\}$$

4. PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) is a population based stochastic optimization technique that was developed by Kennedy and Eberhart (1995). In PSO, each solution is a bird in the flock and is referred to as a particle (Shi and Eberhart, 1998). The PSO has been applied successfully to a wide variety of optimization problems to find optimal or near-optimal solutions. Due to the complexity of

the proposed model, it is very difficult to obtain the optimum solution for this kind of problem by means of traditional approaches. Therefore, in this paper, we apply the PSO to large sized problems.

The general scheme of the applied PSO is provided in Figure 4. In the PSO, the position of the *i*th particle , x_i , is adjusted by a stochastic velocity, V_i . At each iteration of the algorithm, x_i and V_i are calculated according to the following equations (Shi and Eberhart, 1998).

$$V_{id} = w.V_{id} + c_1.rand().(p_{id} - x_{id}) + c_2.Rand().(p_{gd} - x_{id}) \quad (16)$$

$$x_{id} = x_{id} + V_{id} \quad (17)$$

$$-V_{\max} \le V_{id} \le V_{\max} \quad (18)$$

Equation (16) calculates a new velocity for each particle based on its previous velocity, the

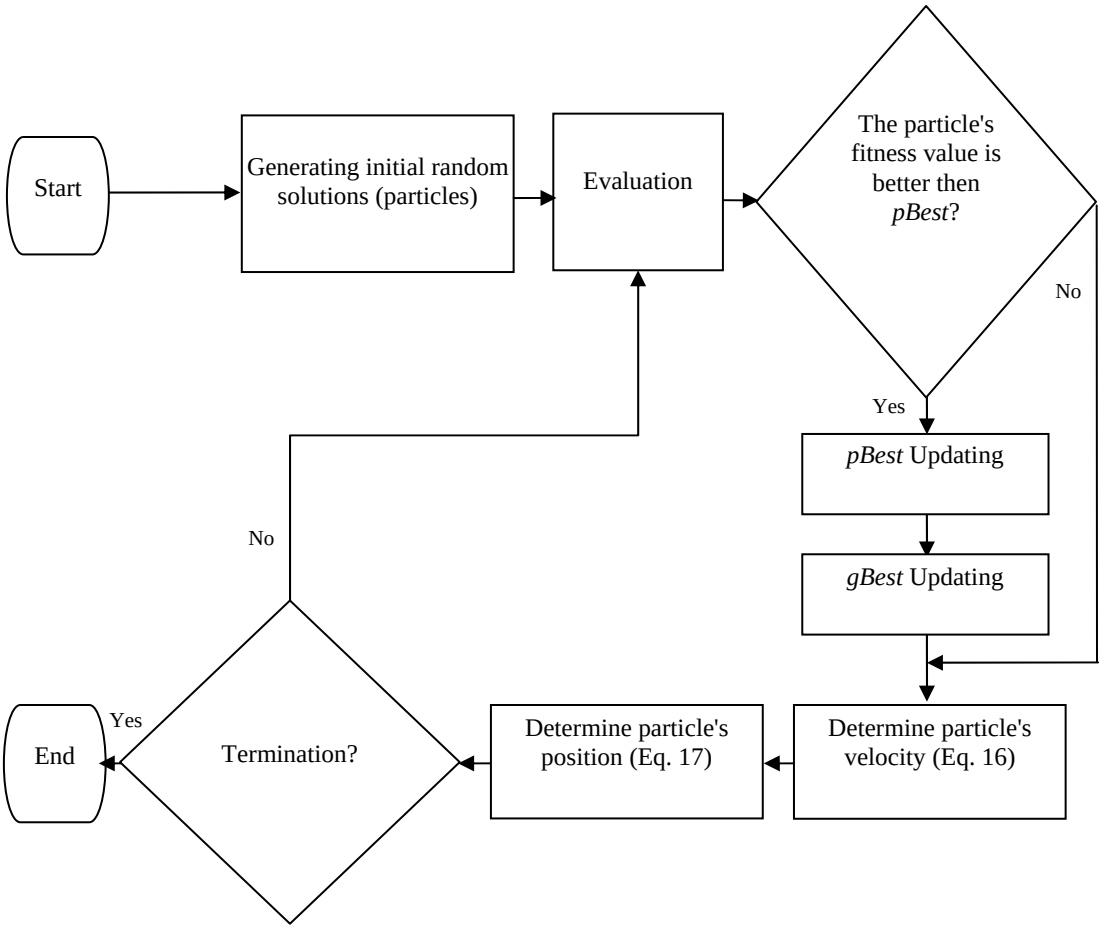

Figure 4. The general scheme of Particle Swarm Optimization algorithm

particle's position at which the best fitness has been achieved so far (p_{id} , or pBest), and the position of the best particle of population (p_{gd} , or gBest). In this equation, Rand () and rand () are two uniformly distributed random numbers in the range [0, 1]. c_1 and c_2 are two learning factors, which control the influence of pBest and gBest on the search process and *w* is an inertia weight that balances the global exploration and local exploitation which has been proposed to decrease linearly with time from a value of 1.4 to 0.5 (Hu, Shi and Eberhart, 2004). $V_{\max}$ is an upper limit on the maximum change of particle velocity.

4.1. Solution Representation

The solution representation should be such that one is able to decode it easily to reduce the cost of implementing the algorithm. In this paper, a continuous representation is used (Tasgetiren et al, 2004).

Table 1. Continuous representation

Generated random numbers	0.36	1.21	2.45	0.59	1.75
Sequence	5	3	1	4	2
Continuous representation	2.45	1.21	0.36	1.75	0.59

To construct the continuous representation, we first need to generate as many random numbers in the range $[0, x_{\max}]$ as the number of jobs, then the smallest of them will be assigned to the position that contains the first job, the next smallest will be assigned to the position that contains the second job, and so on (Table 1.).

5. EXPERIMENTAL RESULTS

To validate the proposed model and the implemented algorithm, various test problems are examined. The algorithm has been coded in Visual Basic 6 and run on Pentium 4 processor with 1.7 GHz and Windows XP using 256 MB of RAM. The experiments are implemented on both small and large-sized problems. For both of these experiments, we consider the following assumptions: 1. the values of ε and δ are equal to 1; 2. each experiment is repeated 15 times; 3. the processing times for all parts on all machines are uniformly generated in the range [10, 100].

Small-sized Problem: The problem instances have been randomly generated as shown in Table 2. The number of particles, termination criterion of PSO, learning factors (c_1 and c_2), and $V_{\max}$ were assumed fixed to be 50, 50, 2, 2, and 3, respectively. For each instance, the results obtained were compared with the ones produced by Lingo8.0. By comparing the results it can be concluded that this algorithm is capable of finding the optimum solutions for most of these problems.

Large-sized Problem: The problem instances were randomly generated as shown in Table 3. The number of particles, termination criterion of PSO, learning factors (c_1 and c_2), and $V_{\max}$ were assumed to be 100, 100, 2, 2, and 3, respectively. Because of the complexity of such problem, the Lingo software cannot produce any results for the large-sized problems. However according to the computational results which is shown in Table 3, this algorithm can achieve suitable solutions in acceptable time.

Table 2. Computational results for small-sized problems

No. of Parts	Problem number	Problem Condition	Lingo 8.0		PSO		
			OFV[a]	Time	OFV		Time
					Ave.	STD[b]	
5	1	$a_i \geq b_i \geq c_i$	483	<1	483	0	<1
	2	$a_i \geq c_i \geq b_i$	435	<1	435	0	<1
	3	$b_i \geq a_i \geq c_i$	363	<1	363	0	<1
	4	$b_i \geq c_i \geq a_i$	459	<1	459	0	<1
	5	$c_i \geq a_i \geq b_i$	454	<1	458	0	<1
	6	$c_i \geq b_i \geq a_i$	404	<1	404	0	<1
	7	Unconditional case	321	<1	323	0	<1
10	8	$a_i \geq b_i \geq c_i$	754	1	754.1	0.3	1
	9	$a_i \geq c_i \geq b_i$	763	1	763	0	1
	10	$b_i \geq a_i \geq c_i$	910	<1	910	0	1
	11	$b_i \geq c_i \geq a_i$	825	1	825	0	1
	12	$c_i \geq a_i \geq b_i$	907	<1	907	0	1
	13	$c_i \geq b_i \geq a_i$	753	<1	753	0	1
	14	Unconditional case	739	132	746.5	6	1
15	15	$a_i \geq b_i \geq c_i$	1312	<1	1312	0	1
	16	$a_i \geq c_i \geq b_i$	1272	<1	1273.4	1.5	1
	17	$b_i \geq a_i \geq c_i$	1212	1	1212.7	0.6	1
	18	$b_i \geq c_i \geq a_i$	1352	<1	1352	0	1
	19	$c_i \geq a_i \geq b_i$	1331	<1	1331	0	1
	20	$c_i \geq b_i \geq a_i$	1222	1	1226.7	4.4	1
	21	Unconditional case	1260	7200[c]	1181.5	13.1	1

a Objective Function Value
b Standard Deviation
c Indicates that the Lingo processing was interrupted after this time and the best achieved value was reported

Table 3. Computational results for large-sized problems

No. of Parts	Problem number	Problem Condition	PSO		
			OFV		Time
			Ave.	STD	
50	22	$a_i \geq b_i \geq c_i$	4427.2	4.5	14.5
	23	$a_i \geq c_i \geq b_i$	4239.1	7.8	14.5
	24	$b_i \geq a_i \geq c_i$	4026.9	11.3	14.5
	25	$b_i \geq c_i \geq a_i$	4334.3	6.8	14.5
	26	$c_i \geq a_i \geq b_i$	4295.1	8.5	14.5
	27	$c_i \geq b_i \geq a_i$	4364.3	2.4	14.5
	28	Unconditional case	3583.1	23.1	14.5
75	29	$a_i \geq b_i \geq c_i$	6347	14.6	24.9
	30	$a_i \geq c_i \geq b_i$	6437.8	8.7	24.9
	31	$b_i \geq a_i \geq c_i$	6469.7	9.6	25.1
	32	$b_i \geq c_i \geq a_i$	6431.6	15.1	25.3
	33	$c_i \geq a_i \geq b_i$	6764.1	8.6	25
	34	$c_i \geq b_i \geq a_i$	6419.4	7.5	25.1
	35	Unconditional case	6173.3	25	25.1
100	36	$a_i \geq b_i \geq c_i$	8889.4	7.6	37.9
	37	$a_i \geq c_i \geq b_i$	8728.5	17	37.8
	38	$b_i \geq a_i \geq c_i$	8836.9	20.6	37.8
	39	$b_i \geq c_i \geq a_i$	8861.9	11.5	37.9
	40	$c_i \geq a_i \geq b_i$	8258.3	15.3	37.9
	41	$c_i \geq b_i \geq a_i$	8645.7	11.3	38.1
	42	Unconditional case	8113.7	30.3	38.1

6. CONCLUSION

In this paper a new mathematical model for a cyclic multiple-part type three-machine robotic cell problem under S^6 robot movement policy was developed that minimizes the cycle time. The developed model is based on Petri nets and provides a new method to calculate cycle times by considering waiting times. It was proved that calculating the cycle time under S^6 policy is unary NP-complete (Hall et al, 1998). Therefore, the PSO algorithm was implemented to tackle the

problem. To validate the model and its solution algorithm, various test problems were considered. The computational results show that the proposed algorithm which is based on PSO could achieve optimum solutions for most small sized problems. For large-sized problems this algorithm was able to find suitable solutions in acceptable time.

REFERENCES

[1] Asfahl C.R. (1992), Robots and manufacturing automation; 2nd Edition, Wiley, New York.

[2] Agnetis A. (2000), Scheduling no-wait robotic cells with two and three machines; *European Journal of Operational Research* 123; 303-314.

[3] Agnetis A., Pacciarelli D. (2000), Part sequencing in three-machine no-wait robotic cells; *Operations Research Letters* 27; 185-192.

[4] Brauner N., Finke G. (1999), On a conjecture about robotic cells: New simplified proof for the three-machine case; *INFOR* 37(1); 20-36.

[5] Bagchi T.P., Gupta J.N.D., Sriskandarajah C. (2006), A Review of TSP based approaches for flow shop scheduling; *European Journal of Operational Research* 169; 816- 854.

[6] Crama Y., Klundert J.J. van de. (1999), Cyclic scheduling in 3-machine robotic flow shops; *Journal of Scheduling* 2; 35-54.

[7] Crama Y., Kats V., Klundert, J.J. van de, Levner E. (2000), Cyclic scheduling in robotic flow shops; *Annals of Operations Research: Mathematics of Industrial Systems* 96; 97-124.

[8] Drobouchevitch I.G., Sethi S.P., Sriskandarajah C. (2006), Scheduling dual gripper robotic cell one-unit cycles; *European Journal of Operational Research* 171; 598-631.

[9] Dawande M., Geismar H.N., Sethi S.P., Sriskandarajah C. (2005), Sequencing and scheduling in robotic cells:Recent developments; *Journal of Scheduling* 8; 387-426.

[10] Gultekin H., Akturk M.S., Karasan O.E. (2006), Cyclic scheduling of a 2-machine robotic cell with tooling constraints; *European Journal of Operational Research* 174; 777–796.

[11] Gultekin H., Akturk M.S., Karasan O.E. (2007), Scheduling in a three-machine robotic flexible manufacturing cell; *Computers & Operations Research* 34; 2463–2477.

[12] Hall N. G., Kamoun H., Sriskandarajah C. (1997), Scheduling in robotic cells: Classification, two and three machine cells; *Operations Research* 45; 421-439.

[13] Hall N. G., Kamoun H., Sriskandarajah C. (1998), Scheduling in robotic cells: Complexity and steady state anhlysis; *European Journal of Operational Research* 109; 43-65.

[14] Hu X., Shi Y., Eberhart R. (2004), Recent advances in particle swarm, in Proc. of CEC2004, Congress on Evolutionary Computation, 1; 90–97.

[15] Kennedy J., Eberhart R. (1995), Particle swarm optimization, in Proc. of the IEEE international conference on neural networks (Perth, Australia), 1942–1948.

[16] Maggot J. (1984), Performance Evaluation of Concurrent Systems Using Petri Nets; *inform. processing lett.*, 18(1); 7-13.

[17] Sriskandarajah C., Hall N.G., Kamoun H., Wan H. (1998), Scheduling large robotic cells without buffers; Annals of Operations Research 76; 287–321.

[18] Sethi S.P., Sriskandarajah C., Sorger G., Blazewicz J., Kubiak W. (1992), Sequencing of parts and robot moves in a robotic cell; International Journal of Flexible Manufacturing Systems 4; 331-358.

[19] Shi Y., Eberhart R. (1998), "A modified particle swarm optimizer", in Proc. of the IEEE international conference on evolutionary computation, 69–73,.

[20] Tasgetiren MF., Sevkli M., Liang YC., Gencyilmaz G. (2004), "Particle swarm optimization algorithm for single machine total weighted tardiness problem". In: Proc. of the IEEE congress on evolutionary computation, Oregon: Portland, 1412-1419.