J. Ind. Eng. Int., 7(14), 41-52, Summer 2011
ISSN: 1735-5702


# Two-machine robotic cell considering different loading and unloading times

*Ali Mohammad Kimiagari [1*]; Hadi Mosadegh [2]*

[1] *Associate Professor, Dept. of Industrial Engineering, Amirkabir University of Technology, Tehran, Iran*

[2] *M.Sc. Student, Dept. of Industrial Engineering, Amirkabir University of Technology, Tehran, Iran*

Received: 5 August 2009; Revised: 13 May 2010; Accepted: 29 May 2010

**Abstract:** In this paper, the researchers have investigated a Concatenated Robot Move (CRM) sequence problem and Minimal Part Set (MPS) schedule problem with different setup times for two-machine robotic cell. They have focused on simultaneous solving of CRM sequence and MPS schedule problems with different loading and unloading times. They have applied a Simulated Annealing (SA) algorithm to provide a good solution rapidly. A domination rule has been developed and used in SA algorithm that strongly reduces the search space and increases speed and solution's quality of the algorithm. Numerical experiments indicate the results of the proposed SA from two points of view: quality of solution and consumed time.

**Keywords:** *CRM sequence; MPS schedule; Robotic cell; Simulated annealing*

## 1. Introduction

Nowadays, robots play an important role in flexible manufacturing systems. Flexible manufacturing systems and just in time systems, widely use robots for manufacturing and producing various types of products. Robots can work in variety of fields such as handling, assemblies and operations. In robotic scheduling literature, also in this paper, the handling robot is considered. Robotic cells consist of one input device, a series of machines, one output device and robots that handle the parts between stations (Dawande *et al.*, 2005). Also there is no buffer storage between machines; therefore, each part at each moment is being processed or blocked on a machine or being handled by robot (Hall *et al.*, 1998). Trying to maximize the throughput rate of cell is a vital decision in competitive environments then in order to achieve this goal, we must minimize the cycle time production. This criterion depends on two main variables: sequence of robot moves and schedule of parts which are going to enter the cell.

Robot move sequence is a set of actions, which robot repeats at each cycle. Loading parts on, or unloading them from a special station, transporting parts from one station to another and all sorts of these activities are robot's duties in the cell. These activities are called Concatenated Robot Move (CRM) sequence.

The part scheduling is the known problem that has been described in classical scheduling problems literature, in which the parts represent the jobs. Some authors investigated this problem for single part type that in this case, we have to solve one of these mentioned problems, i.e. the robot move sequence problem must be solved (Brauner, 2008; Crama and Van de Klundert, 1996; Hall *et al.*, 1997; Sethi *et al.*, 1992).

On the contrary, some of the researchers have had different studies. They determined the robot move cycle at first and solved the scheduling problem of parts which must enter the cell (see Sethi *et al.*, 1992; Chen *et al.*, 1997). Of course many of them concentrated on simultaneous solving of these two problems to optimize the cycle time (Hall *et al.*, 1998; Hall *et al.*, 1997; Aneja and Kamoun, 1999). In this study, the researchers are going to solve these two problems in robotic cell with two machines and multiple part types, simultaneously. They consider one-unit robot move cycle in the problem. The one-unit robot move cycle refers to the state that at each cycle, one part could be completed (Hall *et al.*, 1998).

Since the automation of production technology has been used, production management and planning is also changed. In flexible manufacturing systems, the families of parts that require similar operations are determined and their required operations are used in a specialized cell (Aneja and Kamoun, 1999). However for each family, there are special demands in the current period, thereby each family contains some of the parts which must be produced on their demands. So we should produce

*Corresponding Author Email: kimiagar@aut.ac.ir
Tel.: +98 2166413034

Minimal Part Set (MPS) of each part at each cycle to meet the demands of period.

An MPS is the smallest possible set of parts having the same contributions as the production in period (Hall *et al.*, 1998), e.g. assume that we should produce 300 units of product A, 250 units of product B and 100 units of product C. Consequently, we should produce 6 units of A, 5 units of B and 2 units of C at each cycle. The schedule of parts can redefine as MPS schedule problem. Therefore we have two problems (CRM sequence and MPS schedule), which are going to be solved simultaneously in two-machine robotic cell. Figure (1) shows a sample of such problem with one robot-centered cell (Logendran and Sriskandarrjah, 1996). In spite of the fact that the polynomial time algorithms are developed which get the optimal solution for solving these problems jointly, these studies considered the problems with equivalent loading and unloading time, but the researchers consider different loading and unloading time in this research. In practice, these parameters are not equivalent. Before and after each operation, the primal setup time is required therefore these times should be added to loading and unloading times respectively. Thus different loading and unloading times are considered.

Also the time required for solving the large-scale problems is considerable. Therefore for solving these problems, simultaneously, the researchers developed a meta-heuristic algorithm (i.e. simulated annealing) in order to solve the problem in a reasonable time. In this algorithm they use a domination rule to determine the best CRM sequence for each schedule of MPS. This rule helps to quickly obtain the output with high quality. This paper is organized as follows:

In Section 2 the literature is reviewed briefly. In Section 3 the problem definition and notations are described. Simulated annealing framework is described in Section 4. In Section 5 the validity of algorithm with numerical result in time consumed and solution quality investigate. Sections 6 and 7 present the future researches and conclusion.

## 2. Literature review

Many researchers have concentrated on robotic scheduling problems. The main related literature review and fundamental concepts about our research can be found in Dawande *et al.* (2005). Moreover, Crama *et al.* (2000) and Brauner (2008) have been reviewed and explained related papers and concepts, about robotic scheduling problems.

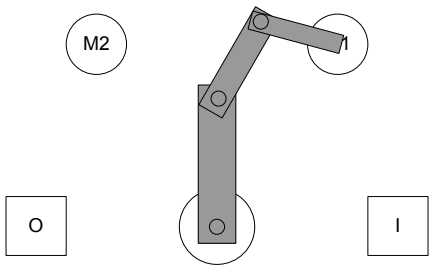


Figure 1: Robotic cell with two-machine.

Ashfal (1985) is one of the famous establishers of robotic flow shop problem. Sethi et al (1992) found the optimal sequence of robot move for two-machine and three-machine cells that produce a single part type and then solved the part scheduling problem for a given robot move sequence, for two-machine cell producing multiple part types. Logendran and Sriskandarajah (1996) introduced three different robotic cell layouts. The in-line robotic cell, the robot-centered cell, and the mobile-robot cell, have been studied in their research. In this paper consider the robot-centered cell problem has been considered.

Hall *et al.* (1997) provided an algorithm with order O ($n^4$) that jointly optimizes the robot move sequence and part scheduling problem in two-machine robotic cell. After two years, Aneja and Kamoun (1999) extended the previous algorithm, and developed an algorithm of complexity O ($n \log n$) that solves the mentioned problem with equivalent load and unload times for all parts on all machines. But in this paper, the researchers are going to solve the problem with different loading and unloading time.

Hall *et al.* (1997) calculated the cycle time for six possible sequence of robot move in three-machine cell. They continued their studies in (Hall *et al.*, 1998) on three-machine cell producing multiple part types and proved that in two out of the six potentially optimal robot move sequence for producing one unit, the part scheduling problem is unary NP-complete.

Crama and Van de Klundert (1996) provided a polynomial time algorithm that optimizes cycle time in $m$ machine cell producing single part type in one robot-centered cell. Chen *et al.* (1997) considered the problem of sequencing parts of different types to minimize the production cycle time when the sequence of the robot moves is given. They used a mathematical formulation for the problem, and then proposed a branch-and-bound algorithm to solve it. Brauner *et al.* (2003) proved that finding the optimal robot move sequence in a robotic cell with general travel time is included in NP-hard class of problems.

Sriskandarajaha *et al.* (1998) categorized the part schedule problem to four groups for $m$ machines in the cell. As a result of this classification, they proved that the part sequencing problems associated with exactly $2m - 2$ of the $m!$ available robot cycles, are solvable in polynomial time. The remaining cycles have associated part sequencing problems which are unary NP-hard.

Gultekin *et al.* (2006) investigated the tooling constraints in a two-machine robotic cell. They solved two problems which are allocating operations to the machines and robot move sequence, simultaneously. They assumed that there are some operations that can be processed only on machine 1 and some operations can be processed only on machine 2. The remaining operations can be processed on either machine which the problem is allocating them to the machines to minimize the cycle time.

Soukhal and Martineau (2005) considered a flow shop robotic cell that processes several jobs. They developed an integer programming model to determine the sequence of jobs that minimizes the makespan criterion. Although their model can solve the problems with $m$ machines, their criterion is makespan and differs from general criterion in cyclic robotic scheduling problems (cycle time). Moreover, their model determines only the schedule of parts.

Brucker and Kampmeyer (2008) presented a general framework for modeling and solving cyclic scheduling problems. Their model covers different cyclic problems such as cyclic job shop problems and cyclic robotic cell problems. They showed that all these problems can be formulated as mixed-integer linear programs which have a common structure.

Recently, Drobouchevitch *et al.* (2010) considered scheduling problem in robotic cell when each machine has one-unit input buffer and one-unit output buffer. In their study, the objective is minimizing robot move sequence or maximizing throughput.

## 3. Problem definition and notations

The main fundamentals of this section are derived from Aneja and Kamoun (1999). The framework of the problem and main notations are borrowed from this reference. To complete description, let's explain the problem and notations here.

We are going to maximize the throughput or minimize the cycle time equivalently. Our robotic cell contains one robot-centered cell, two-machine, one input device and one output device. There is no buffer storage between the machines and if the processing of the part finished, it must be blocked on the machine. In most of such problems, loading and unloading time for any parts are considered equivalently, but we have considered different loading and unloading time for all parts. Travel time, are constant and depend on a fixed parameter.

### 3.1. Problem notations

The problem notations are as follows:

| | |
|---|---|
| $K$ | Number of part types. |
| $P_k$ | The part type $k$, which must be produced, where $k = 1,2,...,K$. |
| $r_k$ | The minimal ratio of part type $k$, at current period where $k = 1,2,...,K$. |
| $MPS$ | In one MPS, $r_k$ parts of type $i$ are produced. $i = 1,...,K$. |
| $n = r_1 + ... + r_k$ | The total number of finished parts in the MPS. |
| $a_i$ | Processing time of part $i$ on machine $M_1$, where $i = 1,2,...,K$. |
| $b_i$ | Processing time of part $i$ on machine $M_2$, where $i = 1,2,...,K$. |
| $I, O$ | Input and output devices respectively. |
| $\varepsilon_i^0$ | Pick up time of part $i$ at $I$ where $i = 1,2,...,K$. |
| $\varepsilon_i^{m1}$ | Loading time of part $i$ on machine $M_m$, where $m = 1,2$ and $i = 1,2,...,K$. |
| $\varepsilon_i^{m2}$ | Unloading time of part $i$ from machine $M_m$, where $m = 1,2$ and $i = 1,2,...,K$. |
| $\varepsilon_i^3$ | Drop time of part $i$ at $O$, where $i = 1,2,...,K$. |
| $\delta$ | Traveling time of robot movement between $I$ to $M_1$, $M_1$ to $M_2$ and $M_2$ to $O$. |
| $2\delta$ | Traveling time of robot movement between $M_2$ to $I$ and $O$ to $M_1$. |
| $3\delta$ | Traveling time of robot movement between $O$ to $I$. |

### 3.2. Problem formulation

Initially, we have to calculate the objective function, i.e. total cycle time of one MPS. At first each partially cycle time between consecutive part, must be considered. This cycle time depends on CRM sequence. For two-machine robotic cell, two possible sequence of robot move are defined by Sethi *et al.* (1992), i.e. $S_1$ and $S_2$. Hall *et al.* (1997) described calculation of these two cycle time (derived by $S_1$ and $S_2$, respectively) in details. We briefly recalculate them by our notations.

Let $\sigma$ be the permutation function which determines the schedule of part $i$. $P_{\sigma(i)}$ indicates that part $i$, is scheduled in the $i^{th}$ position of $\sigma$, where $i=1,...,n$. Also we define $T^s_{\sigma(i)\sigma(i+1)}$ as the time between loading part $P_{\sigma(i)}$ on machine $M_2$ and loading part $P_{\sigma(i+1)}$ on machine $M_2$ by CRM sequence $S_s$, where $s=1,2$ suppose that initial state, starts when part $P_{\sigma(i)}$ is loaded on machine $M_2$. The robot moves order in sequence $S_1$, is as follows: wait when part $P_{\sigma(i)}$ is being processed on $M_2$, unload, move to $O$, drop part $P_{\sigma(i)}$, move to $I$, pick up part $P_{\sigma(i+1)}$, move to $M_1$, load, wait when part $P_{\sigma(i+1)}$ is being processed on machine $M_1$, unload, move to $M_2$ and load. Now according to the mentioned order we can calculate $T^1_{\sigma(i)\sigma(i+1)}$ as follows:

$$\begin{aligned} T^1_{\sigma(i)\sigma(i+1)} &= b_{\sigma(i)} + \varepsilon^{22}_{\sigma(i)} + \delta + \varepsilon^{3}_{\sigma(i)} + 3\delta + \\ &\quad \varepsilon^{0}_{\sigma(i+1)} + \delta + \varepsilon^{11}_{\sigma(i+1)} + a_{\sigma(i+1)} + \\ &\quad \varepsilon^{12}_{\sigma(i+1)} + \delta + \varepsilon^{21}_{\sigma(i+1)} \\ &= 6_{\sigma} + \varepsilon^{0}_{\sigma(i+1)} + \varepsilon^{11}_{\sigma(i+1)} + \varepsilon^{12}_{\sigma(i+1)} + \\ &\quad \varepsilon^{21}_{\sigma(i+1)} + \varepsilon^{22}_{\sigma(i)} + \varepsilon^{3}_{\sigma(i)} + a_{\sigma(i+1)} + b_{\sigma(i)} \end{aligned} \quad (1)$$

We define $w^m_i$ as the waiting time of the robot before unloading part $i$ from machine $M_m$, if necessary. Again, consider initial state, which starts after loading part $P_{\sigma(i)}$ on machine $M_2$. The robot moves order in sequence $S_2$, is as follows: move to $I$, pick up part $P_{\sigma(i+1)}$, move to $M_1$, load, move to $M_2$, wait for processing part $P_{\sigma(i)}$ if necessary, unload, move to $O$, drop part $P_{\sigma(i)}$, move to $M_1$, wait for processing part $P_{\sigma(i+1)}$ if necessary, unload, move to $M_2$ and load. Thus according to the above order, $T^2_{\sigma(i)\sigma(i+1)}$ could be calculated as the following equation:

$$\begin{aligned} T^2_{\sigma(i)\sigma(i+1)} &= 2\delta + \varepsilon^{0}_{\sigma(i+1)} + \delta + \varepsilon^{11}_{\sigma(i+1)} + \delta + \\ &\quad w^2_{\sigma(i)} + \varepsilon^{22}_{\sigma(i)} + \delta + \varepsilon^{3}_{\sigma(i)} + 2\delta + \\ &\quad w^1_{\sigma(i+1)} + \varepsilon^{12}_{\sigma(i+1)} + \delta + \varepsilon^{21}_{\sigma(i+1)} \\ &= 8\delta + \varepsilon^{0}_{\sigma(i+1)} + \varepsilon^{11}_{\sigma(i+1)} + \varepsilon^{12}_{\sigma(i+1)} + \\ &\quad \varepsilon^{21}_{\sigma(i+1)} + \varepsilon^{22}_{\sigma(i)} + \varepsilon^{3}_{\sigma(i)} + w^1_{\sigma(i+1)} + w^2_{\sigma(i)} \end{aligned} \quad (2)$$

where

$$w^2_{\sigma(i)} = \max\left\{0, b_{\sigma(i)} - 4\delta - \varepsilon^{0}_{\sigma(i+1)} - \varepsilon^{11}_{\sigma(i+1)}\right\}$$

Thereby, $w^2_{\sigma(i)}$ is positive if $b_{\sigma(i)}$ is greater than the required time for loading the next part on machine $M_1$ and coming back to $M_2$. $w^1_{\sigma(i+1)}$ could be calculated as following equation:

$$w^2_{\sigma(i)} = \max\left\{0, a_{\sigma(i+1)} - w^2_{\sigma(i)} - 4\delta - \varepsilon^{22}_{\sigma(i)} - \varepsilon^{3}_{\sigma(i)}\right\}$$

Also $w^1_{\sigma(i+1)}$ is positive if $a_{\sigma(i+1)}$ is greater than required time for unloading the previous part from machine $M_2$ and return to $M_1$. Here, we calculate term $w^1_{\sigma(i+1)} + w^2_{\sigma(i)}$ and then extend Equation (2).

$$\begin{aligned} w^1_{\sigma(i+1)} + w^2_{\sigma(i)} &= \max\left\{0, a_{\sigma(i+1)} - w^2_{\sigma(i)} - 4\delta \right. \\ &\quad \left. - \varepsilon^{22}_{\sigma(i)} - \varepsilon^{3}_{\sigma(i)}\right\} + w^2_{\sigma(i)} \\ &= \max\left\{w^2_{\sigma(i)}, a_{\sigma(i+1)} - 4\delta - \varepsilon^{22}_{\sigma(i)} - \varepsilon^{3}_{\sigma(i)}\right\} \\ &= \max\left(\max\left\{0, b_{\sigma(i)} - 4\delta - \varepsilon^{3}_{\sigma(i+1)} \right.\right. \\ &\quad \left.\left. - \varepsilon^{11}_{\sigma(i+1)}\right\}, a_{\sigma(i+1)} - 4\delta - \varepsilon^{22}_{\sigma(i)} - \varepsilon^{3}_{\sigma(i)}\right) \\ &= \max\left\{0, b_{\sigma(i)} - 4\delta - \varepsilon^{0}_{\sigma(i+1)}, a_{\sigma(i+1)} - 4\delta \right. \\ &\quad \left. - \varepsilon^{22}_{\sigma(i)} - \varepsilon^{3}_{\sigma(i)}\right\} \end{aligned}$$

Thus $T^2_{\sigma(i)\sigma(i+1)}$ could be formulated as equation (3).

$$\begin{aligned} T^2_{\sigma(i)\sigma(i+1)} &= 8\delta + \varepsilon^{0}_{\sigma(i+1)} + \varepsilon^{11}_{\sigma(i+1)} + \varepsilon^{12}_{\sigma(i+1)} + \\ &\quad \varepsilon^{21}_{\sigma(i+1)} + \varepsilon^{22}_{\sigma(i+1)} + \varepsilon^{3}_{\sigma(i+1)} + \\ &\quad \max\left\{0, b_{\sigma(i)} - 4\delta - \varepsilon^{0}_{\sigma(i+1)} - \varepsilon^{11}_{\sigma(i+1)}, \right. \\ &\quad \left. a_{\sigma(i+1)} - 4\delta - \varepsilon^{22}_{\sigma(i)} - \varepsilon^{3}_{\sigma(i)}\right\} \end{aligned} \quad (3)$$

### 3.3. Domination rule

Now, the two partially cycle time of two possible CRM sequences, i.e. $T^1_{\sigma(i)\sigma(i+1)}$ and $T^2_{\sigma(i)\sigma(i+1)}$, have been computed. Assuming predetermined MPS schedule, the parts should be entered into cell by one of these sequences. The type of CRM sequence drastically affects on partial cycle time, while the sum of partial cycle times is equal to the total cycle time of MPS.

$$TC = \sum_{i=1}^{n} PC_{\sigma(i)\sigma(i+1)} \tag{4}$$

where $TC$ is the total cycle time and $TC_{\sigma(i)\sigma(i+1)}$ is the partial cycle time between consecutive parts $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$ (i.e. one of these: $T^1_{\sigma(i)\sigma(i+1)}$ or $T^2_{\sigma(i)\sigma(i+1)}$). Note that $P_{\sigma(n+1)}$ is the part which begins the MPS cycle, i.e. $P_{\sigma(1)}$. If the MPS schedule is predetermined, we can find the optimal total cycle time using domination rule for all of consecutive coupled parts $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$.

**Theorem 1:** Assuming the given MPS schedule, for any consecutive parts such as $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$,

If $a_{\sigma(i+1)} + b_{\sigma(i)} < 2\delta$ then

$$T^1_{\sigma(i)\sigma(i+1)} < T^2_{\sigma(i)\sigma(i+1)}$$

**Proof:** We calculate the $T^1_{\sigma(i)\sigma(i+1)} - T^2_{\sigma(i)\sigma(i+1)}$ difference.

$$\begin{aligned} &T^1_{\sigma(i)\sigma(i+1)} - T^2_{\sigma(i)\sigma(i+1)} = \\ &\quad a_{\sigma(i+1)} + b_{\sigma(i)} - \max\left\{0, b_{\sigma(i)} - 4\delta - \varepsilon^0_{\sigma(i+1)} - \right. \\ &\quad \left. \varepsilon^{11}_{\sigma(i+1)}, a_{\sigma(i+1)} - 4\delta - \varepsilon^{22}_{\sigma(i)} - \varepsilon^3_{\sigma(i)} \right\} - 2\delta \\ &\rightarrow T^1_{\sigma(i)\sigma(i+1)} - T^2_{\sigma(i)\sigma(i+1)} = \\ &\quad \min\left\{a_{\sigma(i+1)} + b_{\sigma(i)}, a_{\sigma(i+1)} + 4\delta + \varepsilon^0_{\sigma(i+1)} - \right. \\ &\quad \left. \varepsilon^{11}_{\sigma(i+1)}, b_{\sigma(i)} + 4\delta + \varepsilon^{22}_{\sigma(i)} + \varepsilon^3_{\sigma(i)} \right\} - 2\delta \\ &\qquad\qquad = a_{\sigma(i+1)} + b_{\sigma(i)} - 2\delta \\ &\xrightarrow{a_{\sigma(i+1)} + b_{\sigma(i)} - 2\delta < 0} T^1_{\sigma(i)\sigma(i+1)} < T^2_{\sigma(i)\sigma(i+1)} \end{aligned}$$

Finally, if the MPS is given, we can find the optimal $TC$, by using this rule. For each consecutive part such as $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$, if $a_{\sigma(i+1)} + b_{\sigma(i)} < 2\delta$, the sequence $S_1$, must be used, i.e. under this condition, after loading part $P_{\sigma(i)}$ on machine $M_2$, robot must wait for processing part $P_{\sigma(i)}$, then transfer it to output device and then move toward input device. Otherwise, the sequence $S_2$, must be used, i.e. after loading part $P_{\sigma(i)}$ on machine $M_2$, at first, robot must move toward part $P_{\sigma(i+1)}$ and load it on machine $M_1$, then return to machine $M_2$. In section 4, domination rule will be successfully used in simulated annealing algorithm.

**Corollary 1:** Assuming the given MPS schedule, for all consecutive parts such as $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$,

if $a_{\sigma(i+1)} + b_{\sigma(i)} < 2\delta$, then

$$\begin{aligned} TC^* = 6n\delta + \sum_{i=1}^{n} \left\{ \left( \varepsilon^0_i + \varepsilon^{11}_i + \varepsilon^{12}_i + \varepsilon^{21}_i \right. \right. \\ \left. \left. + \varepsilon^{22}_i + \varepsilon^3_i \right) + \left( a_i + b_i \right) \right\} \end{aligned}$$

**Proof:** for any consecutive parts,

$$\begin{aligned} &PC^*_{\sigma(i)\sigma(i+1)} = T^1_{\sigma(i)\sigma(i+1)} \\ &\rightarrow TC^* = \sum_{i=1}^{n} T^1_{\sigma(i)\sigma(i+1)} = 6n\delta + \sum_{i=1}^{n} \left\{ \left( \varepsilon^0_i + \varepsilon^{11}_i + \right. \right. \\ &\qquad \left. \left. \varepsilon^{12}_i + \varepsilon^{21}_i + \varepsilon^{22}_i + \varepsilon^3_i \right) + \left( a_i + b_i \right) \right\} \end{aligned}$$

## 4. Simulated Annealing

Although there are rapid computer devices which can solve many large-scale problems, many problems exist with infinite needed solution time. These problems are categorized into NP-Hard and NP-Complete class of problems. The scheduling problems are usually NP-Hard, but many researchers have tried to develop polynomial algorithms for some types of these problems. Simultaneous solving of CRM sequence and MPS schedule problems in two-machine robotic cell with equivalent loading and unloading time has been solved in polynomial time by Aneja and Kamoun (1999). Our problem is similar to their problem,

unless the loading and unloading times are different.

Simulated annealing is one of the well-known meta-heuristics algorithms, which is very sensitive on tuning. Because of searching many solutions with very high speed, it is used in this research to solve the problem. Figure 5 show that if the SA is well-tuned, it will converge to the best solution

### 4.1. Search space

We simply can enumerate the total number of possible states. The total possible schedules multiplying by the possible CRM sequences for each MPS schedule, gives the total search space. It means that number of possible solutions which could be investigated by SA algorithm equals to: $n! \times 2^n$.

Using domination rule, we can reduce the search space to $n!$ possible MPS schedule. This rule will be very useful in finding the best CRM sequence for each predetermined MPS schedule rapidly. Therefore, the speed and solution's quality of the algorithm would be strongly increased. Domination rule is completely described in section 3.

However, when the size of the problem increases, the time taken by the polynomial algorithms will be increased naturally. In this section we use simulated annealing algorithm to quickly solve such problems with large scales. In this algorithm we have to jointly optimize two problems: CRM sequence and MPS schedule.

### 4.2. Solution representation

Initially, a random solution would be generated. According to domination rule, only representation of schedule of parts would be required. Another problem i.e. CRM sequence, could be solved by domination rule. In this algorithm, the schedule of parts is represented as a vector, e.g. suppose our MPS contains $n$ parts. One of the $n!$ possible schedules could be represented as Figure 2. Diminsion of this vector is $1\times(n+1)$ and the first part in the vector, is repeated at the end of vector, because in cyclic programming, the MPS cycle must be repeated more times. For any solution, there exists a partner vector which determines the CRM sequence for all parts. This would be considered by domination rule. Dimension of partner vector is $1\times n$ . Figure 3 shows a sample of partner for schedule in Figure 2.

Figure 3 reveals how the parts should enter into cell, e.g. after part 2, part 6 is entered by sequence $S_1$.

### 4.3. Fitness function

The total cycle time is used to evaluate the fitness function. Therefore, the domination rule must be investigated. Moreover, for any solution, two computations would be performed.

At first, each of sequential paired parts is considered. For each consecutive parts $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$, if $a_{\sigma(i+1)} + b_{\sigma(i)} < 2\delta$, then part $P_{\sigma(i+1)}$ enters in cell with sequence $S_1$.

It means that for computing partially cycle time between parts $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$, $T^1_{\sigma(i)\sigma(i+1)}$, which is calculated in section 3, must be considered, otherwise, $T^2_{\sigma(i)\sigma(i+1)}$. The total cycle time would be calculated using equation (4), i.e. sum of partially cycle times between sequential parts, for determined MPS schedule, gives the objective function.

### 4.4. Neighbourhood search and movement

Neighbourhood search could be performed simply and fast. For neighbourhood search, two parts are selected randomly and their places are swapped. This exchange usually changes the fitness function. If new solution is better than the previous one, the algorithm selects a new seed; otherwise, the algorithm calculates the acceptance probability for new bad solution by the following equation:

$$pr = \exp\left(\frac{TC_{old} - TC_{new}}{0.9\times T}\right) \quad (5)$$

If acceptance probability is greater than $0.8$, the new seed will be selected. In Equation (5), $T$ is the temperature. At each temperature, just one iteration is run and cooling schema is as Equation (6).

$$T_{new} = 0.9995 \times T_{old} \quad (6)$$

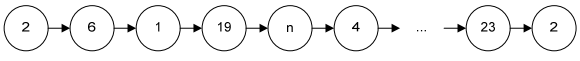


Figure 2: MPS schedule.

Using this schema, the algorithm could search most of possible solutions at high temperatures, however it could search around the optimal solution at low temperatures more carefully. The main framework of tuned simulated annealing using domination rule is presented in Figure 4.

## 5. Numerical experiments

SA algorithm can solve the problem rapidly, but does not guarantee the optimal solution. However, domination rule, described in section 3, helps to find the solutions rapidly with high quality. To quickly examine validation of the algorithm, we changed the transportation time ($\delta$) for different sizes of the problem. When $\delta$ value increased, the number of CRM sequence $S_1$ increased too, and by decreasing $\delta$ value, the number of $S_2$ sequence, increased.

### 5.1. Domination rule verification

In this section the researchers compared two types of SA algorithms by numerical experiments.

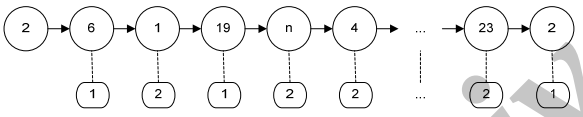


Figure 3: MPS schedule and CRM sequence.

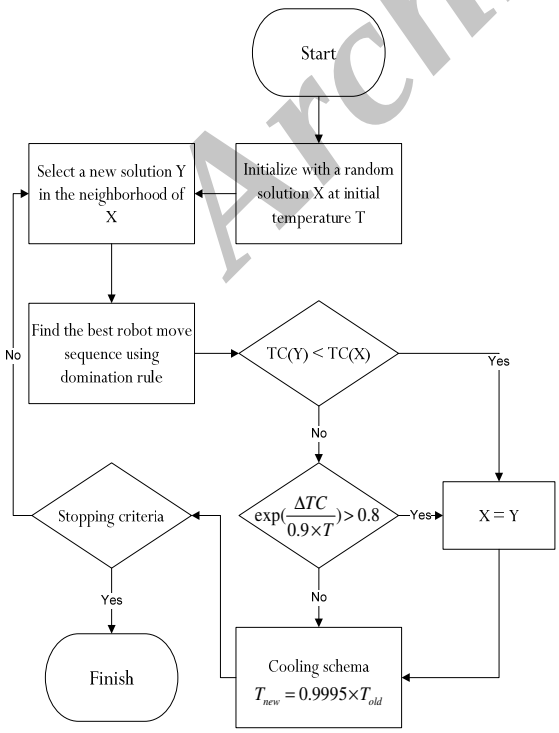


Figure 4: Dominated SA algorithm flowchart.

One of these algorithms employs domination rule (DSA), and another one, without the mentioned rule, searches all of $n!\times 2^n$ possible solutions. Framework of the second algorithm is similar to ordinary SA algorithm (OSA). In OSA algorithm, solution representation is similar to Figure 3, where for implementing partner vector neighbourrhood search, two sequences should be randomly selected and exchanged. After exchanging, one random-selected sequence should be replaced with number 1 or 2 with probability 50%.

Because these mentioned algorithms do not guarantee the optimality, an exact method is necessary. Blind-search algorithms will be useful for small-sized problems.

Therefore, the exact solution could be found by enumerating all possible $n!$ schedules using domination rule. The general steps of blind-search algorithm are presented below:

Step 1: calculate all possible $n!$ schedules.

Step 2: for each schedule, find the best CRM sequence using domination rule (Theorem 1) presented in Section 3.

Step 3: calculate the total cycle time for each MPS schedule and its CRM sequence which obtained in step 2, using Equation (4) presented in Section 3.

Step 4: find the minimum cycle time for all $n!$ solutions as the best solution.

To examine how the OSA and DSA algorithms work, some random test problems are generated and solved. Table 1 compares three algorithms for small-sized problems: blind-search, DSA and OSA. For each algorithm, solution and time are two throughputs presented in Table 1.

Table 1 shows that both of OSA and DSA algorithms have good performance for small-sized problems, where all the gaps for DSA are 0% and for one case, OSA algorithm has 0.3% difference with exact solution. According to Table 1, time of blind-search algorithm will be increased exponentially. Random data, which generated for problems in Table (1), are presented in Tables 5, 6 and 7, respectively.

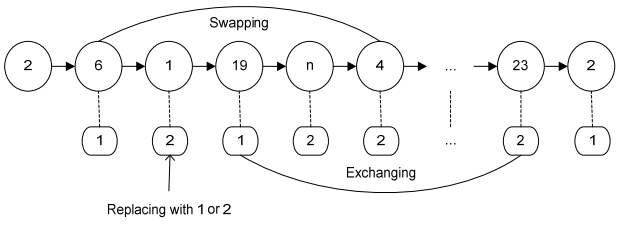


Figure 5: Three neighbourhood search schemas for OSA algorithm; swapping for MPS schedule, exchanging and replacing for CRM sequence.

Processing time on machine 1 and 2 is between 1 to 100 and 1 to 200 respectively, which are given in Table (5). The data in Table (6) (loading and unloading time) are between 1 to 41, and the data in Table (7) (robot travelling time) are between 1 to 101. These data are generated in the mentioned intervals uniformly. All of the algorithms are run by MALAB on a computer with Intel C2D 2GHz CPU and RAM 2GB.

Due to the intolerable time required using exact methods such as blind-search algorithm for large-sized problems, meta-heuristics algorithms are selected instead. However, another way to examine the superiority of the domination rule is comparing DSA versus OSA for solving more large-sized problems.

A statistical analysis is applied in order to demonstrate the superiority of the proposed approach. Therefore, a randomized complete block design is used. The general statistical model for both time and solution is as follows:

$$Y_{ij} = \mu + Size_i + A\lg orithm_j + \varepsilon_{ij} \quad (7)$$

Where $Size_i$ represents the block effect (problem size), then $A\lg orithm_j$ stands for two mentioned algorithms (OSA and DSA). Variable $Y_{ij}$ is throughput of the experiment which can be one of the above mentioned criteria. After selecting size of the problem randomly, two algorithms have been run five times in a random order. The obtained solution and the time consumed are recorded in Table 2. The behavior of two algorithms is rather same for small problems, but they are different in the consumed time and quality of solution when the size is increased. To statistically analyze the data of Table 2, the SAS software is used for the two criteria. The results of Table 3 show that the difference between the algorithms, regarding quality of solutions, is significant.

Table 4 also shows DSA and OSA outputs, attending to the time consumed. Obviously, the time required for solving large-scale problems will be strongly significant between DSA algorithm and OSA algorithm, because in OSA algorithm the search space is very large, rather than DSA algorithm, which its search space is reduced using the domination rule.

According to Tables 2, 3 and 4 three consequences are presented below:

1. In large-scales, the DSA algorithm can obtain better solutions than OSA algorithm.
2. Time required for DSA algorithm is significantly low comparing with OSA algorithm.
3. The two types of algorithm almost converge to the optimal solution.

Figure 6 shows that how DSA algorithm behaves at size 50. At first, the algorithm has many alternatives to move, then due to the cooling schema, it will be converged to the best solution.

Finally, according to the above results, we can employ dominated simulated annealing algorithm when we have to solve large-scale problems, requiring time saving and good solution.

## 6. Future researches

For future works we suggest following researches:

A. Considering the uncertainties in the robotic cell problems. Probabilistic or fuzzy times for loading and unloading operation can be investigated for two or more machines.

B. Developing an exact model for solving robotic scheduling problem with two-machines with different loading and unloading time.

C. Developing a same research for solving CRM sequence and MPS schedule problems simultaneously in robotic cell with three machines.

Table 1: DAS and OSA algorithms verification.

| Problem Size | Exact Solution | | Dominated SA (DSA) | | | Ordinary SA (OSA) | | |
|---|---|---|---|---|---|---|---|---|
| | Solution | Time (s) | Solution | Time (s) | Gap (%) | Solution | Time (s) | Gap (%) |
| 3 | 1838 | 0 | 1838 | 0.2 | 0.0% | 1838 | 0.2 | 0.0% |
| 4 | 3203 | 0 | 3203 | 0.2 | 0.0% | 3203 | 0.3 | 0.0% |
| 5 | 2030 | 0 | 2030 | 0.4 | 0.0% | 2030 | 0.6 | 0.0% |
| 6 | 4081 | 0 | 4081 | 0.4 | 0.0% | 4081 | 0.8 | 0.0% |
| 7 | 5292 | 0.1 | 5292 | 0.5 | 0.0% | 5292 | 0.9 | 0.0% |
| 8 | 6722 | 0.7 | 6722 | 0.6 | 0.0% | 6722 | 0.9 | 0.0% |
| 9 | 7320 | 3.4 | 7320 | 0.6 | 0.0% | 7340 | 1 | 0.3% |
| 10 | 8018 | 51.8 | 8018 | 0.7 | 0.0% | 8018 | 1 | 0.0% |

Table 2: Comparison of DSA and OSA algorithms.

| Problem Size | Ordinary SA (OSA) | | Dominated SA (DSA) | |
|---|---|---|---|---|
| | Solution | Time (s) | Solution | Time (s) |
| 3 | 1133 | 8 | 1133 | 1 |
| | 1133 | 6.9 | 1133 | 1 |
| | 1133 | 7.3 | 1133 | 1 |
| | 1133 | 8.9 | 1133 | 1 |
| | 1133 | 7.6 | 1133 | 1 |
| 5 | 2311 | 8.5 | 2311 | 1.2 |
| | 2311 | 8.7 | 2311 | 1 |
| | 2311 | 8.6 | 2311 | 1 |
| | 2311 | 8.6 | 2311 | 1.1 |
| | 2311 | 8.7 | 2311 | 1 |
| 10 | 6814 | 12.7 | 6779 | 1.5 |
| | 6805 | 13 | 6779 | 1.9 |
| | 6805 | 13.9 | 6779 | 1.3 |
| | 6823 | 13.8 | 6779 | 1 |
| | 6842 | 12.5 | 6779 | 2 |
| 15 | 6926 | 15.7 | 6619 | 2.4 |
| | 7012 | 15.4 | 6619 | 2 |
| | 6945 | 13.7 | 6619 | 2.1 |
| | 6934 | 14.5 | 6619 | 1.9 |
| | 6995 | 15.7 | 6619 | 2.5 |
| 20 | 8142 | 25.7 | 7874 | 5.2 |
| | 8103 | 26.3 | 7874 | 5.2 |
| | 8091 | 27.6 | 7874 | 5.5 |
| | 8038 | 28.6 | 7874 | 5.2 |
| | 8152 | 25.4 | 7874 | 5.3 |
| 25 | 5423 | 39.9 | 4880 | 5.8 |
| | 5298 | 36.9 | 4880 | 5.9 |
| | 5320 | 37.8 | 4880 | 5.4 |
| | 5276 | 38 | 4880 | 5.5 |
| | 5284 | 35.6 | 4880 | 5.6 |
| 30 | 17344 | 42.2 | 16937 | 5.4 |
| | 17890 | 44.1 | 16937 | 5.6 |
| | 17465 | 45.8 | 16937 | 5.1 |
| | 18030 | 42.3 | 16937 | 5.7 |
| | 17268 | 44.4 | 16937 | 5.4 |
| 35 | 21300 | 39.1 | 20856 | 5.1 |
| | 21109 | 42.4 | 20856 | 4.2 |
| | 21117 | 38.7 | 20856 | 5.9 |
| | 21094 | 47.3 | 20856 | 5.7 |
| | 21100 | 46.6 | 20856 | 6.1 |
| 40 | 22430 | 42.3 | 22027 | 4.8 |
| | 22427 | 50.9 | 22027 | 5.2 |
| | 22655 | 46.6 | 22027 | 6.5 |
| | 22498 | 48.7 | 22027 | 4.8 |
| | 22794 | 49.9 | 22027 | 5.9 |
| 50 | 70171 | 61.5 | 58274 | 7.1 |
| | 62962 | 66.4 | 58262 | 5.4 |
| | 64104 | 59.8 | 58265 | 6.5 |
| | 65152 | 57.1 | 58261 | 6.7 |
| | 70654 | 62.1 | 58302 | 6.2 |

Table 3: SAS results for quality of solution using data of Table 2.

| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| Model | 10 | 29820616540 | 2982061654 | 1337.29 | <.0001 |
| Error | 89 | 198463352 | 2229925 | | |
| Corrected Total | 99 | 30019079893 | | | |
| | R-Square | Coeff Var | Root MSE | Y Mean | |
| | 0.993389 | 9.752082 | 1493.293 | 15312.56 | |
| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
| Algorithm | 1 | 29558794 | 29558794 | 13.26 | 0.0005 |
| Size | 9 | 29791057746 | 3310117527 | 1484.41 | <.0001 |

Table 4: SAS results for time consumed using data of Table 2.

| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| Model | 10 | 27358.40460 | 2735.84046 | 37.29 | <.0001 |
| Error | 89 | 6529.42290 | 73.36430 | | |
| Corrected Total | 99 | 33887.82750 | | | |
| | R-Square | Coeff Var | Root MSE | Y Mean | |
| | 0.807322 | 49.81272 | 8.565296 | 17.19500 | |
| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
| Algorithm | 1 | 17580.10810 | 17580.10810 | 239.63 | <.0001 |
| Size | 9 | 9778.29650 | 1086.47739 | 14.81 | <.0001 |

Table 5: Random processing time of parts on each machine related to Table 1.

| Problem Size | Machine Time | Parts | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 3 | a | 45 | 89 | 2 | | | | | | | |
| | b | 132 | 103 | 164 | | | | | | | |
| 4 | a | 97 | 59 | 54 | 25 | | | | | | |
| | b | 38 | 143 | 170 | 169 | | | | | | |
| 5 | a | 10 | 60 | 71 | 9 | 92 | | | | | |
| | b | 91 | 33 | 74 | 187 | 155 | | | | | |
| 6 | a | 88 | 85 | 27 | 73 | 10 | 99 | | | | |
| | b | 108 | 171 | 150 | 91 | 176 | 192 | | | | |
| 7 | a | 58 | 8 | 66 | 51 | 25 | 76 | 1 | | | |
| | b | 52 | 59 | 14 | 37 | 164 | 164 | 147 | | | |
| 8 | a | 66 | 80 | 82 | 60 | 55 | 94 | 84 | 73 | | |
| | b | 161 | 13 | 159 | 141 | 16 | 108 | 93 | 84 | | |
| 9 | a | 41 | 88 | 68 | 68 | 13 | 51 | 46 | 3 | 10 | |
| | b | 43 | 145 | 56 | 100 | 43 | 77 | 48 | 127 | 110 | |
| 10 | a | 5 | 34 | 13 | 39 | 67 | 96 | 38 | 67 | 53 | 50 |
| | b | 47 | 55 | 195 | 74 | 52 | 45 | 172 | 118 | 98 | 43 |

Table 6: Random loading and unloading time at each station related to Table 1.

| Problem Size | I | M1 | | M2 | | O |
|---|---|---|---|---|---|---|
| | | load | unload | load | unload | |
| | 0 | 11 | 12 | 21 | 22 | 3 |
| 3 | 19 | 20 | 40 | 15 | 36 | 20 |
| | 13 | 13 | 25 | 28 | 9 | 8 |
| | 19 | 39 | 28 | 19 | 29 | 33 |
| 4 | 32 | 23 | 13 | 25 | 11 | 33 |
| | 25 | 15 | 31 | 29 | 30 | 3 |
| | 21 | 13 | 37 | 5 | 35 | 37 |
| | 20 | 6 | 2 | 3 | 28 | 23 |
| 5 | 20 | 26 | 26 | 26 | 12 | 2 |
| | 14 | 33 | 3 | 16 | 33 | 25 |
| | 16 | 39 | 23 | 9 | 19 | 35 |
| | 17 | 6 | 5 | 28 | 23 | 28 |
| | 3 | 37 | 33 | 30 | 30 | 3 |
| 6 | 36 | 27 | 13 | 27 | 24 | 19 |
| | 15 | 12 | 17 | 39 | 3 | 11 |
| | 10 | 18 | 37 | 8 | 17 | 23 |
| | 20 | 28 | 2 | 17 | 18 | 10 |
| | 4 | 33 | 32 | 16 | 14 | 20 |
| | 36 | 15 | 7 | 41 | 32 | 20 |
| 7 | 26 | 4 | 15 | 21 | 8 | 5 |
| | 9 | 36 | 35 | 29 | 24 | 25 |
| | 27 | 32 | 40 | 28 | 25 | 34 |
| | 40 | 24 | 21 | 4 | 23 | 4 |
| | 34 | 13 | 37 | 7 | 10 | 15 |
| | 35 | 2 | 8 | 28 | 7 | 27 |
| | 31 | 7 | 39 | 17 | 28 | 33 |
| 8 | 8 | 38 | 3 | 37 | 31 | 8 |
| | 36 | 10 | 23 | 7 | 24 | 29 |
| | 25 | 16 | 33 | 7 | 14 | 4 |
| | 23 | 30 | 18 | 14 | 16 | 16 |
| | 8 | 15 | 8 | 25 | 23 | 33 |
| | 33 | 34 | 7 | 24 | 22 | 13 |
| | 11 | 20 | 29 | 11 | 31 | 24 |
| | 39 | 19 | 14 | 10 | 36 | 39 |
| 9 | 15 | 18 | 10 | 22 | 31 | 24 |
| | 13 | 24 | 7 | 6 | 25 | 26 |
| | 23 | 20 | 23 | 15 | 9 | 36 |
| | 16 | 21 | 11 | 6 | 38 | 5 |
| | 17 | 5 | 10 | 39 | 13 | 34 |

Table 6: Random loading and unloading time at each station related to Table 1 (continued).

| Problem Size | I | M1 | | M2 | | O |
|---|---|---|---|---|---|---|
| | | load | unload | load | unload | |
| | 0 | 11 | 12 | 21 | 22 | 3 |
| 10 | 2 | 7 | 12 | 36 | 16 | 40 |
| | 1 | 40 | 8 | 23 | 37 | 26 |
| | 13 | 20 | 37 | 23 | 18 | 21 |
| | 35 | 40 | 15 | 28 | 11 | 30 |
| | 33 | 3 | 31 | 25 | 24 | 39 |
| | 13 | 40 | 22 | 36 | 35 | 33 |
| | 32 | 27 | 39 | 35 | 23 | 2 |
| | 37 | 4 | 2 | 23 | 5 | 15 |
| | 40 | 21 | 40 | 25 | 28 | 25 |
| | 9 | 20 | 25 | 36 | 9 | 23 |
| | 27 | 40 | 25 | 35 | 24 | 33 |
| | 17 | 17 | 18 | 29 | 26 | 29 |
| | 1 | 25 | 3 | 22 | 12 | 3 |
| | 20 | 14 | 16 | 14 | 27 | 10 |

Table 7: The travelling times of robot movement between $I$ to$M_2$, $M_2$ to $M_2$ and $M_2$ to $O$ ( $\hat{O}$ ).

| Problem Size | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|
| travelling time | 60 | 88 | 36 | 71 | 86 | 94 | 95 | 91 |

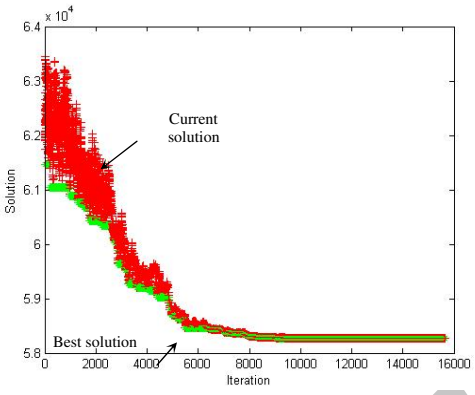


Figure 6: DSA behavior for size 50.

## 7. Conclusion

The researchers investigated the robotic scheduling problem with two-machine which defined by Hall *et al.* (1997). They concentrated on setup times, i.e. different loading and unloadding time for all parts. For simultaneous solving two related problems (CRM sequence and MPS schedule), three algorithms have been applied: blind-search, dominated SA (DSA) and ordinary SA (OSA). Because of using domination rule in DSA, the numerical experiments showed the superiority of DSA against other algorithms, especially in large-sized problems, from quality of solution and time consumed observations.

## References

Aneja, Y. P.; Kamoun, H., (1999), Scheduling of parts and robot activities in a two machine robotic cell. *Computers & Operations Research*, 26(4), 297-312.

Asfahl, C. R., (1985), *Robots and manufacturing automation*. John Wiley & Sons, New York, NY.

Brauner, N., (2008), Identical part production in cyclic robotic cells: Concepts, overview and open questions. *Discrete Applied Mathematics*, 156(13), 2480-2492.

Brauner, N.; Finke, G.; Kubiak, W., (2003), Complexity of one-cycle robotic flow shops. *Journal of Scheduling*, 6(4), 355-371.

Brucker, P.; Kampmeyer, T., (2008), A general model for cyclic machine scheduling problems. *Discrete Applied Mathematics*, 156(13), 2561-2572.

Chen, H.; Chu, C.; Proth, J. M., (1997), Sequencing of parts in robotic cells. *International Journal of Flexible Manufacturing Systems*, 9(1), 81-104.

Crama, Y.; Katsb, V.; Van de Klundert, J.; Levner, E., (2000), Cyclic scheduling in robotic flow shops. *Annals of Operations Research*, 96, 97-124.

Crama, Y.; Van de Klundert, J., (1996), Cyclic scheduling of identical parts in a robotic cell. *Operations Research*, 45(6), 952-965.

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.

Drobouchevitch, I. G.; Geismar, H. M.; Sriskandarajah, C., (2010), Throughput optimization in robotic cell with input and output machine buffers: A Comparative study of two key

models. *European Journal of Operational Research*, 206(3), 623-633.

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(2), 777-796.

Hall, N. G.; Kamoun, H.; Sriskandarajah, C., (1997), Scheduling in robotic cells: classification, two and three-machine cells. *Operations Research*, 45(3), 421-439.

Hall, N. G.; Kamoun, H.; Sriskandarajah, C., (1998), Scheduling in robotic cells: complexity and steady state analysis. *European Journal of Operational Research*, 109(1), 43-63.

Logendran, R.; Sriskandarrjah, C., (1996), Sequencing of robot activities and parts in two-machine robotic cells. *International Journal of Production Research*, 34, 3447-3463.

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(3), 331-358.

Sriskandarajaha, C.; Hall, N. G.; Kamoun, H., (1998), Scheduling large robotic cells without buffers. *Annals of Operations Research*, 76, 287-321.

Soukhal, A.; Martineau, P., (2005), Resolution of a scheduling problem in a flow shop robotic cell. *European Journal of Operational Research*, 161(1), 62-72.