International Journal of Production Research

ISSN: 0020-7543 (Print) 1366-588X (Online) Journal homepage: http://www.tandfonline.com/loi/tprs20

Scheduling in robotic cells: process flexibility and cell layout

Hakan Gultekin , M. Selim Akturk & Oya Ekin Karasan

To cite this article: Hakan Gultekin , M. Selim Akturk & Oya Ekin Karasan (2008) Scheduling in robotic cells: process flexibility and cell layout, International Journal of Production Research, 46:8, 2105-2121, DOI: 10.1080/00207540601100262

To link to this article: http://dx.doi.org/10.1080/00207540601100262

International Journal of Production Research,
Vol. 46, No. 8, 15 April 2008, 2105–2121

Taylor & Francis
Taylor & Francis Group

Scheduling in robotic cells: process flexibility and cell layout

HAKAN GULTEKIN, M. SELIM AKTURK* and
OYA EKIN KARASAN

Department of Industrial Engineering, Bilkent University, 06800 Bilkent,
Ankara, Turkey

(*Revision received February 2006*)

The focus of this study is the identical parts robotic cell scheduling problem with m machines under the assumption of process and operational flexibility. A direct consequence of this assumption is a new robot move cycle that has been overlooked in the existing literature. We prove that this new cycle dominates all classical robot move cycles considered in the literature for $m=2$. We also prove that changing the layout from an in-line robotic cell to a robot-centered cell reduces the cycle time of the proposed cycle even further, whereas the cycle times of all other cycles remain the same. For the m-machine case, we find the regions where the proposed cycle dominates the classical robot move cycles, and for the remaining regions present its worst case performance with respect to classical robot move cycles. Considering the number of machines as a decision variable, we also find the optimal number of machines that minimizes the cycle time of the proposed cycle.

Keywords: Robotic cell; Cyclic scheduling; Automated manufacturing; Cell layout

1. Introduction

A manufacturing cell in which loading and unloading operations are performed by robots is called a robotic cell. Three different cell layouts have been examined in the literature: robot-centered cells (where the robot movement is rotational), in-line robotic cells (where the robot moves linearly), and mobile-robot cells (generalization of in-line robotic cells and robot-centered cells) (Logendran and Sriskandarajah 1996). The existing robotic cell scheduling literature considers in-line or mobile robotic cells. In this study we initially consider the in-line robotic cell layout. An in-line robotic cell with m machines is shown in figure 1. However, as Han and Cook (1998) stated, layout analysis can improve the efficiency of the cells. In particular, Mata and Tubaileh (1998) discuss the machine layout problem, i.e. the locations and orientations of the machines in a flexible manufacturing cell served by a single robot. It is generally known that robot-centered cells are preferred

*Corresponding author. Email: akturk@bilkent.edu.tr

International Journal of Production Research
ISSN 0020–7543 print/ISSN 1366–588X online
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540601100262

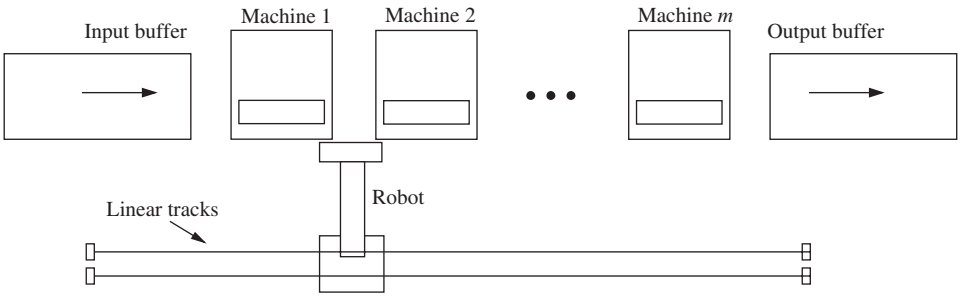

Figure 1. In-line robotic cell layout.

in practice because they reduce the required physical space. We will analytically demonstrate that changing the robotic cell layout from an in-line to a robot-centered cell can improve the effectiveness of these systems.

In this study we consider m-machine robotic cells that repeatedly produce identical parts where each part consists of a finite set of operations. The objective is the minimization of the cycle time, which is equivalent to maximizing the throughput. Since the robot follows a computer program, there must be a finite activity sequence for the robot that it repeats to produce the parts. Therefore, the robot activities must be cyclic and minimizing this cycle time is a relevant objective. In a robotic cell for machining operations, the processing stations are predominantly CNC machines and these machines can communicate with the robot as well as with the cell controller on a real-time basis. CNC machines possess operational and process flexibility by definition. *Operational flexibility* is defined as the capability of changing the ordering of several operations, whereas *process flexibility* is defined as the capability of performing several operations at the same machine (Browne *et al*. 1996). As a consequence, a part is assumed to be composed of a set of operations, each having an individual operation time and requiring a different cutting tool. These cutting tools are stored in the tool magazines of the CNC machines. Whenever an operation requires a different tool, the tool change can be done easily and in a very short period of time, provided that the required tool is available in the tool magazine. Consequently, each CNC machine is capable of performing all operations of a given part as long as it has the required tools in its tool magazine.

In the current robotic cell scheduling literature, each part passes through the machines in the same order and the processing time of each part on each machine is a known parameter, P_i, for machine $i = 1, 2, \ldots, m$. In some industries, such as electroplating and plastic molding, since the parts must follow the same sequence of operations, this assumption is valid. However, in metal cutting industries, in which highly flexible CNC machines are used, Akturk *et al*. (2005) proved that considering the allocation of operations to machines as decision variables can improve the efficiency of the cells. Assuming operation allocation decisions are predetermined on each machine limits the number of alternatives unnecessarily and overlooks the flexibility of CNC machines. In this study we will assume that all of the machines are loaded with at least one copy of all of the required tools. As a consequence, each machine is capable of performing all of the operations of each part and, hence, each operation can be performed by any of the machines. The processing time of the parts on each machine depends on the allocation of the operations

to the machines. Different allocations yield different processing times on machines, thus different cycle time values. Additionally, although the parts are identical and have the same set of operations, the allocation of operations may be different for each part. Unlike the current literature on identical parts robotic cell scheduling, the problem is not only to find the optimal robot move sequence, but also to decide on the machine to process each operation of each part (allocation of operations to the machines) that jointly minimizes the cycle time.

The main purpose of this study is to propose a new robot move cycle that fully utilizes the operational and process flexibility of CNC machines. This cycle will then be compared with the classical robot move cycles present in the literature. Interestingly, this new cycle is used extensively in industry, not because it has been proved to be optimal, but because it is very practical, and easy to understand and implement. In this study, we prove that this cycle is not only simple and practical but also dominates all classical robot move cycles when there are two machines. For the general m-machine case, we provide the regions where the proposed cycle dominates the classical robot move cycles and for the remaining regions we analyse the worst case performance of the proposed cycle with respect to classical robot move cycles. In addition to this operational problem, we also answer the particular design problem of determining the optimum number of CNC machines served by a single robot required to minimize the cycle time.

In the literature, these systems are assumed to be flow-shop-type systems in which each part to be processed passes through the input buffer (M_0), the first machine (M_1) through to the mth machine (M_m) in respective order, and, finally, the output buffer ($M_{(m+1)}$). There are no buffers at or between the machines and the robot and the machines can hold one part at a time. The state of the system is defined by the location of the robot and whether the robot and the machines are loaded or empty. After loading a part to one of the machines, the robot either waits for the part to complete its processing or moves to unload another machine. (There is no waiting time when taking a part from the input buffer or dropping a part to the output buffer.)

In this study we consider identical parts. For the complexity of the multiple parts case we refer the reader to the works of Sriskandarajah *et al.* (1998) and Hall *et al.* (1998). The necessary framework for this problem was initially developed by Sethi *et al.* (1992). They showed that, in an m-machine robotic cell, there are $m!$ one-unit cycles, where an n-unit cycle is defined as a sequence of robot moves in which each machine is loaded and unloaded exactly n times and the cell returns to its initial state. One-unit cycles are attractive since they are practical and easy to understand and control. Sethi *et al.* (1992) also proved that, for a two-machine robotic cell producing a single part type, the optimal solution is a one-unit cycle, and conjectured that *optimal one-unit cycles are superior to every n-unit cycle for* $n \geq 2$. Crama and Van de Klundert (1997) considered the identical parts problem with m machines and showed that, considering only one-unit cycles, the problem can be solved in (strongly) polynomial time. Hall *et al.* (1997) considered three machine cells producing single part types and proved that the repetition of one-unit cycles dominates more complicated policies that produce two units. The validity of the conjecture of Sethi *et al.* (1992) for three-machine robotic flow shops was established by Crama and Van de Klundert (1999). Brauner and Finke (1999) showed that the conjecture is not valid for $m \geq 4$.

In the next section we define the problem more formally and introduce the notation and definitions pertinent to this study. In section 3 we analyse the two-machine case. Section 4 is devoted to the analysis of the m-machine case. Finally, section 5 concludes the paper and suggests future research directions.

2. Problem definition

In this section we give a formal definition of our problem and introduce the basic terminology and notation. We shall adopt the following definition borrowed from Crama and Van de Klundert (1997).

Definition 2.1: Robot activity $\mathbf{A}_i$ consists of the following moves of the robot: unload a part from machine i, transport it to machine $i+1$, and load machine $i+1$.

According to this definition, in an m-machine robotic cell we have exactly $m+1$ robot activities, $A_0, A_1, \ldots, A_m$, where the machines are numbered $1, 2, \ldots, m$, the input buffer is numbered 0 and the output buffer is numbered $m+1$. Since, in an optimal cycle, we require that the robot move path is as short as possible, any two consecutive activities uniquely determine the robot moves between them. Therefore, any robot move cycle can be uniquely described by a permutation of the above activities. Additionally, Crama *et al.* (2000) make the following basic feasibility assumptions, which we shall also incorporate into our study.

(1) The robot cannot load an already loaded machine.
(2) The robot cannot unload an already unloaded machine.

These assumptions restrict the ordering of the activities. For example, for two machines we have only two feasible one-unit robot move cycles:

$$S1\colon\ A_0A_1A_2,$$
$$S2\colon\ A_0A_2A_1.$$

In this study we assume that all of the operations of a part can be allocated to only one machine and that each machine has the capability of performing all of the operations. This problem can also be viewed as a parallel machine scheduling problem with a common server, as described by Hall *et al.* (2000). The operational and process flexibility of CNC machines allows the possibility of new cycles, which necessitates definitions of new robot activities: let A_{0i} be the robot activity in which the robot takes a part from the input buffer and loads machine $i = 1, 2, \ldots, m$, and let $A_{i(m+1)}$ be the robot activity in which the robot unloads machine i and drops the part to the output buffer, where $i = 1, 2, \ldots, m$.

In an m-machine robotic cell there are exactly $2m$ activities. Using these activities we can define new cycles as follows.

Definition 2.2: Under a *pure cycle*, starting with an initial state, the robot performs each of the $2m$ activities ($A_{0i}, A_{i(m+1)}$, $i = 1, \ldots, m$) exactly once and the final state of the system is identical to the initial state.

Note that, under these cycles, all of the operations of each part are performed completely by one of the machines, and between two loadings of any one machine,

all other machines are loaded exactly once. One repetition of such a cycle produces m parts and in order to find the cycle time (long-run average time to produce one part) we divide the total time necessary to complete one repetition of this cycle by m. Each permutation of the $2m$ activities defines a pure cycle. However, some permutations define the same pure cycle. For example, in the two-machine case, $A_{01}A_{02}A_{13}A_{23}$ and $A_{13}A_{23}A_{01}A_{02}$ are different representations of the same cycle. As a result, after eliminating the different representations, there exist a total of $(2m-1)!$ different pure cycles in an m-machine cell. With this many different pure cycles, finding the best and later comparing it with all the classical flow shop type robot move cycles is extremely cumbersome and hence is omitted from the scope of the current paper. Instead, we focus on the simplest and most widely used pure cycle as a representative of this huge class. We prove that even this cycle dominates all classical robot move cycles for two-machine cells and performs very well for general m-machine cells. The proposed cycle is defined by the following activity sequence for m machines.

Definition 2.3 $A_{01}A_{02}\ldots A_{0m}A_{1(m+1)}A_{2(m+1)}\ldots A_{m(m+1)}$: the robot first loads machines 1 through m with a different part in respective order and each machine starts processing all of the operations of its loaded part. Then, the robot unloads machines 1 through m respectively. In order to unload machine i, the robot returns back to machine i, waits in front of the machine if the processing of the part is not finished, unloads the machine, transports the part to the output buffer and drops the part.

The cycle time derivation of the proposed cycle is presented in appendix A. We will use the parameters and decision variables shown in table 1.

In the next section we will focus on the two-machine case and show the dominance of the proposed cycle over the traditional robot move cycles.

3. Two-machine case

In this section we compare the cycle times of the proposed cycle and the traditional robot move cycles. The following definition will be used throughout this study.

Table 1. Parameters and decision variables.

o_i	processing time of operation i. Note that the processing times of operation i on all machines are equal, $\forall i=1,2,\ldots,r$, where r is the number of operations necessary to produce one part
P	total processing time of the operations to be allocated to the machines, i.e. $P=\sum_{i=1}^{r} o_i$
ϵ	the load and unload time of workstations by the robot
δ	time taken by the robot to travel between any two adjacent stations. We assume this time to be additive. That is, the time required for the robot to move from machine i to machine j is the sum of the movement times between all of the intervening pairs of machines in the route from machine i to j where $i, j \in [1,2,\ldots,m]$
T	long-run average cycle time to produce one part

Definition 3.1: We define an allocation pattern to be a specific allocation of the operations to the m machines. A cycle using k different allocation patterns means that the allocation of every kth part in the infinite sequence is identical and k is minimal with this property.

Following this definition, let P_{ij} be the total processing time of the part with allocation pattern j on machine i. This is equivalent to the summation of the processing times of the operations that are allocated to machine i according to allocation pattern j.

The following theorem derives a lower bound for the cycle time of any robot move cycle in the m-machine case for which the system is assumed to be a flow shop.

Theorem 3.2: *For an m-machine flow-shop-type robotic cell, the cycle time of any n-unit cycle is no less than*

$$\underline{T_{fs(m)}} = \max\{2(m+1)(\epsilon+\delta) + \min\{P, \delta\}, 4\epsilon + 4\delta + (P/m)\}. \tag{1}$$

Proof: Geismar *et al*. (2005) derived the following lower bound for classical robot move cycles when there is no flexibility, i.e. the allocation and the ordering of the operations are assumed to be fixed and known for each machine:

$$\max\left\{2(m+1)(\epsilon+\delta) + \sum_{i=1}^{m} \min\{P_i, \delta\}, 4\epsilon + 4\delta + \max_i\{P_i\}\right\}, \tag{2}$$

where P_i is the processing time on machine i. The reasoning behind the first argument of the max function in (2) is as follows. The robot loads and unloads all m machines exactly once ($2m\epsilon$), and also takes a part from the input buffer (ϵ), and drops a part to the output buffer (ϵ) in every cycle, resulting in $2(m+1)\epsilon$. As forward movement, the robot travels all the way from the input buffer to the output buffer in a sequence of robot activities that takes at least $(m+1)\delta$, and in order to return to the initial state, the robot must travel back to the input buffer, taking at least $(m+1)\delta$. Additionally, note that each loading operation is followed by an unloading operation of either the same or a different machine. (Note that, taking a part from the input buffer is assumed to be an unloading operation.) The summation term in the first argument of (2) represents the total time between all loading and the subsequent unloading operations. After loading a part to machine i, the robot has the following options: it either waits in front of the machine, awaiting the completion of the processing of the part before unloading it (P_i), or travels to another machine to unload it or travels to the input buffer to take another part. The minimum travel time from machine i to any other machine is δ. Thus, for machine i, in order to find a lower bound we take the minimum of these two values and for all m machines this totals $\sum_{i=1}^{m} \min\{P_i, \delta\}$. With the assumptions of this study, the total robot travel time and load/unload time do not change. However, the sum term in (2) follows from the fact that the processing times on the machines are fixed. In this study, the processing times are not fixed but depend on allocation patterns, in other words, they are decision variables. For a cycle with k different allocation patterns where k is arbitrary, the cycle is repeated k times, each repetition with differing processing times. After loading a part to machine i, the robot either waits in front of the machine, awaiting completion of processing (P_{ik}), or travels to another machine to unload it or to the input buffer to take a part, which takes at least δ time. Hence, for

all machines and for all repetitions of the cycle we have $\sum_{j=1}^{k}\sum_{i=1}^{m}\min\{P_{ij},\delta\}$. In order to find the lower bound to produce one part we must divide this by k. Furthermore, $\min\{P,\delta\} \le \sum_{i=1}^{m}\min\{P_{ij},\delta\}$ for any allocation j, since $\sum_{i=1}^{m} P_{ij} = P$. Then we have $(1/k)\sum_{j=1}^{k}\min\{P,\delta\} = \min\{P,\delta\}$. As a consequence, with the assumptions of this study, the first argument of the max function reduces to

$$2(m+1)(\epsilon+\delta)+\min\{P,\delta\}.$$

The reasoning behind the second argument of the max function in (2) is the following. The cycle time of any cycle is greater than the time between two consecutive loadings of a machine for which the consecutive loading time is the greatest. But in order to make a consecutive loading, the robot must at least perform the following activities. After loading a part to some machine i, the minimum time required before unloading this part is P_i. Then, the robot unloads machine i (ϵ), transports the part to machine $(i+1)$ (δ), loads it (ϵ), returns to machine $(i-1)$ (2δ), unloads it (ϵ), transports the part to machine i (δ) and loads it (ϵ). This gives in total $4\epsilon+4\delta+P_i$. In order to find the greatest consecutive loading time we take $\max_i\{P_i\}$. However, with the assumption of process and operational flexibility, for a cycle with k different allocation patterns, where k is arbitrary, the longest processing time is $\max_{i,j}(P_{ij})$. Since $\sum_{i=1}^{m} P_{ij} = P$, $\forall i$, we have $P/m \le \max_{i,j}(P_{ij})$. Hence, with the assumptions of this study, the second argument of the max function reduces to $4\epsilon+4\delta+P/m$. This completes the proof. □

Using (A1) the cycle time of the proposed cycle with $m=2$ becomes

$$T_{\text{proposed}(2)} = 4\epsilon+6\delta+1/2\max\{0,P-(2\epsilon+4\delta)\}, \tag{3}$$

and using (1) the lower bound for the traditional robot move cycles becomes

$$\underline{T_{fs(2)}} = \max\{6\epsilon+6\delta+\min\{P,\delta\},4\epsilon+4\delta+P/2\}. \tag{4}$$

The following theorem will establish an important contribution of this paper.

Theorem 3.3: *The proposed robot move cycle $A_{01}A_{02}A_{13}A_{23}$ gives the minimum cycle time for the two-machine identical parts robotic cell scheduling problem with process and operational flexibility.*

Proof: A simple comparison of equations (3) and (4) for $P\in[0,\delta]$, $P\in(\delta,2\epsilon+4\delta]$, $P\in(2\epsilon+4\delta,4\epsilon+6\delta]$ and $P\in(4\epsilon+6\delta,\infty)$ yields $T_{\text{proposed}(2)} \le \underline{T_{fs(2)}}$. □

Note that the proposed cycle is not necessarily the best pure cycle. However, theorem 3.3 proves that even this cycle dominates all of the classical robot move cycles. In a two-machine cell there are six pure cycles, $C1$ through $C6$, for which the activity sequences and the cycle time values are presented in appendix B. The following theorem makes a comparison among the pure cycles and determines the regions of optimality.

Theorem 3.4: *If $P<2\epsilon+4\delta$, $C1$ is optimal; if $P>2\epsilon+4\delta$, $C6$ is optimal; if $P=2\epsilon+4\delta$, both $C1$ and $C6$ perform equally well.*

Proof: Observing the cycle times of the cycles presented in appendix B, one can readily conclude that $C1$ dominates $C2$, $C3$, $C4$ and $C5$. A simple comparison of the cycle times of $C1$ and $C6$ concludes the proof. □

At the beginning of this study, we assumed an in-line robotic cell layout (IRC). For this layout we proved that, if we assume operational and process flexibility, the new cycle gives better results than all of the common cycles reported in the literature. At this point we consider changing the layout of the cell to a robot-centered one (RCC), which is shown in figure 2. Although Han and Cook (1998) stated that layout analysis can improve the efficiency of the cells, the classical robotic cell scheduling literature does not compare the cycle times of robot move cycles with IRC and RCC layouts. This is due to the fact that, for the common cycles reported in the literature, both layout types give the same cycle time, assuming that, for both types of cell layout, the robot transportation time between two adjacent machines is fixed at δ and assumed to be additive. In the robot-centered cell layout, as seen in figure 2, the travel time from the input buffer to machine 1 or machine 2 is δ and the travel time from machine 1 to machine 2 is equivalent to the summation of travel times from machine 1 to the input buffer (or output buffer) (δ) and from the input buffer (output buffer) to machine 2 (δ), which gives 2δ. The travel times for the IRC and RCC layouts are different. For example, the travel time from machine 1 to machine 2 is δ in the IRC layout, whereas it is 2δ in the RCC layout. As a consequence, the cycle time of the proposed cycle will be different for these two layouts. In the following theorem we compare the cycle times of the proposed cycle with the IRC and RCC layouts and prove that the cycle time with the RCC layout is less than the cycle time with the IRC layout.

Theorem 3.5: *For two-machine robotic cells, the cycle time of the proposed cycle with the RCC layout is less than the cycle time with the IRC layout.*

Proof: First, let us derive the cycle time of the proposed cycle with the RCC layout. Initially, the machines are empty and the robot is in front of the input buffer. The robot takes a part (ϵ), transports it to the first machine (δ), loads it (ϵ), returns to the input buffer (δ), takes another part (ϵ), transports it to the second machine (δ), loads it (ϵ), returns to the first machine (2δ), waits if necessary

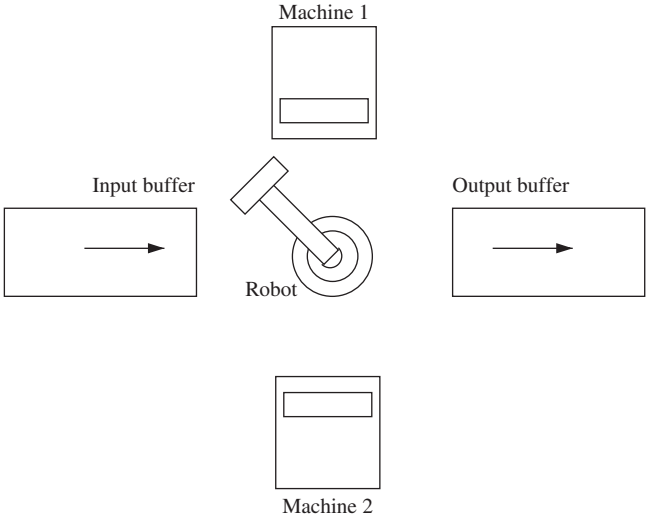

Figure 2. Robot-centered cell layout.

for the machine to finish processing of the part (w_1), unloads the machine (ϵ), transports the part to the output buffer (δ), drops it (ϵ), moves back to the second machine (δ), waits if necessary (w_2), unloads the machine (ϵ), transports the part to the output buffer (δ), drops it (ϵ) and returns to the input buffer (2δ). Note that, during one cycle, two parts are produced. Thus, in order to find the cycle time we divide the total time by 2, giving

$$T_{\text{proposed(RCC)}} = 4\epsilon + 5\delta + w_1 + w_2,$$

where $w_1 = \max\{0, P - (2\epsilon + 4\delta)\}$ and $w_2 = \max\{0, P - (2\epsilon + 4\delta + w_1)\}$ and hence $w_1 + w_2 = \max\{w_1, P - (2\epsilon + 4\delta)\} = \max\{0, P - (2\epsilon + 4\delta)\}$. Consequently, the cycle time of the proposed cycle with the RCC layout is

$$T_{\text{proposed(RCC)}} = 4\epsilon + 5\delta + \frac{1}{2}\max\{0, P - (2\epsilon + 4\delta)\}.$$

On the other hand, the cycle time for the proposed cycle with the IRC layout is given by (3). After a simple comparison we conclude that changing the layout proves to be favorable for the proposed cycle. □

The above theorem is important since, in many practical applications, robot-centered cells are used simply because a particular type of cellular layout requires less space than an in-line robotic cell layout. Furthermore, stationary base robots (as in robot-centered cells) are cheaper to install and easier to program and, consequently, more robust than mobile robots. As a final remark, in the new move cycle, each part is loaded and unloaded only once, which means less gaging, one probable reason why this cycle is preferred in practice. In the next section we consider the *m*-machine case.

4. The *m*-machine case

For two machines it is proved that, although the proposed cycle is not the optimal robot move cycle in all regions, it dominates over all of the classical robot move cycles. Additionally, a comparison is made among the pure cycles and the regions of optimality for these cycles are determined. However, since the number of pure cycles increases significantly as the number of machines increases, finding the best pure cycle is a huge enumerative task for $m \geq 3$. Henceforth, we will only compare the proposed cycle with the classical robot move cycles. Recall that the proposed cycle is a direct consequence of assuming the machines to be CNC machines that are loaded with at least one copy of each of the required tools. Since it is easy to control and implement, such a cycle is preferred in industry to more complex cycles, even if it is not the provably optimal robot move cycle.

Following the conclusions drawn in the previous section, one can conjecture that the proposed cycle is always optimal and there is no reason to consider the robot move cycles derived under the assumption of a flow-shop-type system. In particular, if the whole processing of a part can be done on a single machine, there is no reason to perform a portion of it on one machine and the rest on another. In this way, some load/unload time will be saved. As we will see below, this conjecture holds when the robot is the bottleneck, that is, when the total

processing time of the parts is small with respect to the load/unload time, ϵ, and transportation time, δ. However, when the machines are the bottleneck instead of the robot, that is, the total processing time is greater with respect to ϵ and δ, the proposed cycle may result in larger cycle time values. If the processing time exceeds a certain value, then the average idle time of the machines waiting for a part to be loaded becomes greater in the proposed cycle. The following three-machine example provides a situation of this kind.

Example 4.1: Let us assume that each part requires six operations with $o_1 = 40$, $o_2 = 45$, $o_3 = 50$, $o_4 = 60$, $o_5 = 50$ and $o_6 = 55$, making the total processing time of each part $P = 300$. Also, let $\epsilon = 2$ and $\delta = 10$. Consider the one-unit cycle $S6$, which is defined by the activity sequence $A_0A_3A_2A_1$. The cycle time for this cycle was derived by Sethi *et al.* (1992) as

$$T_{S6} = 8\epsilon + 12\delta + \max\{0, a - 4\epsilon - 8\delta, b - 4\epsilon - 8\delta, b - 4\epsilon - 8\delta\},$$

where a, b and c are the processing times on M_1, M_2 and M_3, respectively. Let us consider the following allocation of operations: operations 1 and 4 are allocated to the first machine, $a = 100$; operations 2 and 6 are allocated to the second machine, $b = 100$; and operations 3 and 5 are allocated to the last machine, $c = 100$. Note that this allocation corresponds to a one-allocation pattern and with our notation $a = P_{11}$, $b = P_{21}$ and $c = P_{31}$. The cycle time in this case is $T_{S6} = 148$. On the other hand, using (A1) with $m = 3$ and with the given data, the cycle time of the proposed cycle is 152.

This example shows that we cannot establish the dominance of the proposed cycle over the traditional robot move cycles for $m \geq 3$. However, the proposed cycle may not be the best pure cycle in this case. For example, consider $A_{01}A_{34}A_{03}A_{24}A_{02}A_{14}$. The cycle time of this cycle with the parameters of the above example turns out to be 129.33. However, there are 120 pure cycles in three-machine cells and finding regions of optimality for these cycles is unnecessarily cumbersome. Hence, in the remainder we will only consider the proposed cycle and prove that even this cycle performs very efficiently.

4.1 *Regions where the proposed cycle dominates the traditional robot move cycles*

With the following theorem we find the regions where the proposed cycle dominates, with respect to cycle time, the traditional robot move cycles for m-machine robotic cells. Recall that section 3 was restricted to the two-machine case. Below we shall consider the $m \geq 3$ case.

Theorem 4.2: *In comparison with the traditional robot move cycles, the proposed cycle is the best if* $(m-2)\delta \leq 2\epsilon$ *or* $P \leq 2(m^2-1)\epsilon + (m^2+2m-2)\delta$.

Proof: In order to prove this theorem, we will compare the cycle time of the proposed cycle with the lower bound value of the traditional robot move cycle times. Let us first recall that

$$\underline{T_{fs(m)}} = \max\left\{2(m+1)(\epsilon+\delta) + \min\{P, \delta\}, 4\epsilon + 4\delta + \left(\frac{P}{m}\right)\right\},$$

and

$$T_{\text{proposed}(m)} = 4\epsilon + 2(m+1)\delta + \frac{1}{m}(\max\{0, P - 2(m-1)\epsilon - (m-1)(m+2)\delta\}).$$

Note that both of these are piecewise functions of P and can be rewritten as follows:

$$\underline{T_{fs(m)}} = \begin{cases} 2(m+1)(\epsilon+\delta) + P, & \text{if } P \le \delta, \\ 2(m+1)\epsilon + (2m+3)\delta, & \text{if } \delta < P \le 2m(m-1)\epsilon + m(2m-1)\delta, \\ 4\epsilon + 4\delta + P/m, & \text{if } P > 2m(m-1)\epsilon + m(2m-1)\delta, \end{cases}$$

$$T_{\text{proposed}(m)} = \begin{cases} 4\epsilon + 2(m+1)\delta, & \text{if } P \le 2(m-1)\epsilon + (m-1) \\ & \quad \times (m+2)\delta, \\ \dfrac{1}{m}\Big(2(m+1)\epsilon & \\ \quad +(m^2+m+2)\delta + P\Big), & \text{if } P > 2(m-1)\epsilon + (m-1) \\ & \quad \times (m+2)\delta. \end{cases}$$

A simple comparison leads to $\delta \le 2(m-1)\epsilon + (m-1)(m+2)\delta \le 2m(m-1)\epsilon + m(2m-1)\delta$. Hence, we will consider the following cases.

(1) If $0 \le P \le \delta$, then the cycle time of the proposed cycle is

$$T_{\text{proposed}(m)} = 4\epsilon + 2(m+1)\delta.$$

The lower bound of the cycle times of traditional robot move cycles is

$$\underline{T_{fs(m)}} = 2(m+1)\epsilon + 2(m+1)\delta + P.$$

Clearly, $T_{\text{proposed}(m)} \le \underline{T_{fs(m)}}$.

(2) If $\delta < P \le 2(m-1)\epsilon + (m-1)(m+2)\delta$, then

$$T_{\text{proposed}(m)} = 4\epsilon + 2(m+1)\delta \le 2(m+1)\epsilon + (2m+3)\delta = \underline{T_{fs(m)}}.$$

(3) If $2(m-1)\epsilon + (m-1)(m+2)\delta < P \le 2m(m-1)\epsilon + m(2m-1)\delta$, then

$$T_{\text{proposed}(m)} = 1/m(2(m+1)\epsilon + (m^2+m+2)\delta + P),$$

and

$$\underline{T_{fs(m)}} = 2(m+1)\epsilon + (2m+3)\delta.$$

When we compare these two values we see that

$$T_{\text{proposed}(m)} \le \underline{T_{fs(m)}} \iff P \le 2(m^2-1)\epsilon + (m^2+2m-2)\delta,$$

which is one of the conditions in the statement of our theorem. Recall that, in this region, $P \le 2m(m-1)\epsilon + m(2m-1)\delta$. If $(m-2)\delta \le 2\epsilon$, then $P \le 2m(m-1)\epsilon + m(2m-1)\delta \le 2(m^2-1)\epsilon + (m^2+2m-2)\delta$. As a result, if $(m-2)\delta \le 2\epsilon$ and $P \le 2m(m-1)\epsilon + m(2m-1)\delta$, then $T_{\text{proposed}(m)} \le \underline{T_{fs(m)}}$.

(4) If $P > 2m(m-1)\epsilon + m(2m-1)\delta$, then

$$T_{\text{proposed}(m)} = 1/m(2(m+1)\epsilon + (m^2+m+2)\delta + P),$$

and

$$\underline{T_{fs(m)}} = 4\epsilon + 4\delta + P/m.$$

Comparing these two, one can show that, for $(m-2)\delta \leq 2\epsilon$, $T_{\text{proposed}(m)} \leq \underline{T_{fs(m)}}$. □

Outside these regions, we can provide a worst case performance bound of the proposed cycle with respect to the traditional robot move cycles. In particular,

Lemma 4.3: *In the region where* $(m-2)\delta > 2\epsilon$ *and* $P > 2(m^2-1)\epsilon + (m^2+2m-2)\delta$, *the cycle time of the proposed cycle is* $T_{\text{proposed}(m)} < C \cdot T^*$, *where* $C = 1 + [(m^2-3m+2)/(m^2+6m-2)]$ *and* T^* *is the optimal cycle time among the traditional robot move cycles.*

Proof: In this region we know from (1) that $T_{fs(m)} \geq 4\epsilon + 4\delta + P/m$. The cycle time of the proposed cycle in this region is

$$T_{\text{proposed}(m)} = 1/m(2(m+1)\epsilon + (m^2+m+2)\delta + P).$$

Hence, we can derive a worst case performance bound for using the proposed cycle instead of the best flow-shop-type robot move cycle as follows. Let T^* be the optimal cycle time among the traditional robot move cycles in this region:

$$\begin{aligned}\frac{T_{\text{proposed}(m)}}{T^*} &\leq \frac{1/m(2(m+1)\epsilon + (m^2+m+2)\delta + P)}{1/m(4m\epsilon + 4m\delta + P)} \\ &= \frac{2(m+1)\epsilon + (m^2+m+2)\delta + P}{4m\epsilon + 4m\delta + P} \\ &= 1 + \frac{-2(m-1)\epsilon + (m-1)(m-2)\delta}{4m\epsilon + 4m\delta + P}.\end{aligned}$$

Since $P > 2(m-1)(m+1)\epsilon + (m^2+2m-2)\delta$, we have

$$\frac{T_{\text{proposed}(m)}}{T^*} < 1 + \frac{-2(m-1)\epsilon + (m-1)(m-2)\delta}{(2m^2+4m-2)\epsilon + (m^2+6m-2)\delta}.$$

Let $\delta = \alpha\epsilon$ where $\alpha > 2/(m-2)$:

$$\frac{T_{\text{proposed}(m)}}{T^*} < 1 + \frac{(m-1)(m-2)\alpha - 2(m-1)}{(m^2+6m-2)\alpha + (2m^2+4m-2)}.$$

The right-hand side becomes larger as α tends to infinity (the loading/unloading time is negligible when compared with the robot transportation time). Hence, the bound converges asymptotically to the following:

$$\begin{aligned}\frac{T_{\text{proposed}(m)}}{T^*} &< \lim_{\alpha\to\infty}\left(1 + \frac{(m-1)(m-2)\alpha - 2(m-1)}{(m^2+6m-2)\alpha + (2m^2+4m-2)}\right) \\ &= 1 + \frac{m^2-3m+2}{m^2+6m-2}.\end{aligned}$$

□

For $m=2$ the worst case bound is 1 and for $m \to \infty$ the asymptotic bound is 2. As a consequence, the worst case bound takes values between 1 and 2 with respect to m. For example, when $m=4$ the worst case bound becomes $1 + 6/38 \approx 1.158$.

4.2 *Determining the optimal number of machines for the proposed robot move cycle*

In the previous section we studied the operational problem of determining the robot move sequences for a given number of machines. Now let us consider the number of machines as a decision variable and try to find the optimal number of machines that minimizes the cycle time for given parameters ϵ, δ and P. The cycle time for the proposed cycle for the most general m-machine case is given by (A1). In the following lemma we show that this function is convex with respect to m.

Lemma 4.4: *The cycle time of the proposed cycle given in* (A1) *is convex with respect to m.*

Proof: We can rewrite this function as

$$\max\left\{2m\delta + 4\epsilon + 2\delta, \frac{1}{m}(m^2\delta + 2m\epsilon + m\delta + P + 2\epsilon + 2\delta)\right\},$$

which is equivalent to

$$\max\left\{2m\delta + 4\epsilon + 2\delta, m\delta + 2\epsilon + \delta + \frac{1}{m}(P + 2\epsilon + 2\delta)\right\}. \tag{5}$$

The first argument of the above max function is linear with respect to m. The second argument is a summation of two convex functions: $m\delta + 2\epsilon + \delta$ and $1/m(P + 2\epsilon + 2\delta)$ (note that $m > 0$). Thus, it is also convex. Finally, the maximum of two convex functions is also a convex function. □

Let a be a real number. We will denote the largest integer smaller than or equal to a by $\lfloor a \rfloor$. The following theorem determines the optimal number of machines given the parameters ϵ, δ and P.

Theorem 4.5: *The optimal number of machines, m*, is one of the two integers* $\lfloor 1/2\delta(-2\epsilon - \delta + \alpha) \rfloor$ *or* $\lfloor 1/2\delta(-2\epsilon - \delta + \alpha) \rfloor + 1$, *where* $\alpha = (4\epsilon^2 + 12\epsilon\delta + 9\delta^2 + 4\delta P)^{1/2}$.

Proof: We are trying to minimize a function of m of the form $f(m) = \max\{g(m), h(m)\}$, where, $g(m) = 2m\delta + 4\epsilon + 2\delta$ and $h(m) = m\delta + 2\epsilon + \delta + 1/m(P + 2\epsilon + 2\delta)$. Let m^* denote the minimizer of $f(m)$. Then m^* satisfies at least one of the following: m^* is a minimizer of $g(m)$, it is a minimizer of $h(m)$ or $g(m^*) = h(m^*)$. Let us consider each of these cases.

(1) $g(m)$ is a linear increasing function and is minimized for $m=0$. However, $h(m)$ tends to ∞ for $m \to 0$. Since $f(m)$ takes the maximum of $g(m)$ and $h(m)$, the minimizer of $g(m)$ cannot be a minimizer of $f(m)$.

(2) $h(m)$ is a convex continuous function for $m > 0$

$$\frac{\partial h(m)}{\partial m} = 0 \Rightarrow \delta - 1/m^2(2\epsilon + 2\delta + P) = 0 \Rightarrow \hat{m} = \sqrt{1/\delta(2\epsilon + 2\delta + P)}.$$

However, at this point,

$$g(\hat{m}) = 2\delta\sqrt{1/\delta(2\epsilon + 2\delta + P)} + 4\epsilon + 2\delta$$
$$> 2\delta\sqrt{1/\delta(2\epsilon + 2\delta + P)} + 2\epsilon + \delta = h(\hat{m}).$$

Hence, the minimizer of $h(m)$ cannot be a minimizer of $f(m)$.

(3) Hence, we can conclude that the minimizer of (5) is at the intersection point of the two arguments of the max function, which is found as follows:

$$g(m) = h(m) \Rightarrow 2m\delta + 4\epsilon + 2\delta = m\delta + 2\epsilon + \delta + \frac{1}{m}(2\epsilon + 2\delta + P)$$
$$\Rightarrow m^2\delta + (2\epsilon + \delta)m - 2\epsilon - 2\delta - P = 0.$$

We can find the roots of this equation by using the discriminant. There are two roots, one of which is less than 0. But since we consider the region where $m > 0$ we take the non-negative root as the solution of this equation:

$$m = 1/2\delta\left(-2\epsilon - \delta + \sqrt{4\epsilon^2 + 4\epsilon\delta + \delta^2 + 4\delta(2\epsilon + 2\delta + P)}\right)$$
$$= 1/2\delta(-2\epsilon - \delta + \alpha),$$

where $\alpha = \sqrt{4\epsilon^2 + 12\epsilon\delta + 9\delta^2 + 4\delta P}$. This is a real number. However, m represents the number of machines, which means it must be an integer. From lemma 4.4, the function is convex with respect to m. As a consequence, in order to find the best integer value we have to consider both sides of the real number. That is, the largest integer smaller than $1/2\delta(-2\epsilon - \delta + \alpha)$ and the smallest integer larger than this number. The best integer value is one of $\lfloor 1/2\delta(-2\epsilon - \delta + \alpha)\rfloor$ or $\lfloor 1/2\delta(-2\epsilon - \delta + \alpha)\rfloor + 1$, where α is defined as before. In order to find which one of these two gives the minimum cycle time value, we evaluate equation (5) at these two integer values and take the one which gives the minimum cycle time value. □

5. Conclusion

In this paper we have considered the m-machine identical parts robotic cell scheduling problem with operational and process flexibility. We proved in theorem 3.4 that a new robot move cycle that is used extensively in industry due to its simplicity to understand and implement, in fact dominates the traditional robot move cycles for $m = 2$. With theorem 4.2 we found the regions where the proposed cycle dominates the traditional robot move cycles for $m \geq 3$. We also found a worst case performance bound of the proposed cycle with respect to the traditional robot move cycles for the remaining regions. An additional contribution of this study is determining the optimal number of machines that minimizes the cycle time of the proposed cycle. Furthermore, with the reduced cycle times (increased throughput), our results enable the justification for the additional tool inventories that will be incurred when loading a copy of every required tool to both of the machines (this might also necessitate a larger tool magazine). Another important contribution of this study comes from the layout analysis. Robot-centered cells require less physical space than in-line robotic

cells and mobile robot cells. In this study, we proved in theorem 3.5 that changing the layout from an in-line robotic cell to a robot-centered cell for $m = 2$ reduces the cycle time of the new cycle even further while all others remain the same.

In theorem 4.2, the regions where the proposed cycle dominates the classical robot move cycles is determined for $m \geq 3$. In order to prove this theorem, we compared the proposed cycle with the lower bound of the classical robot move cycles. For the remaining regions, Gultekin *et al*. (2006) proved that the proposed cycle dominates all but one of the one-unit robot move cycles and all of the two-unit robot move cycles in three-machine cells. However, extending these results to $m \geq 4$ machine cells is an open problem. Note that one-unit cycles need not be optimal and the allocation of operations to the machines for each robot move cycle is to be determined, an additional increase in the difficulty of the problem. Finally, instead of identical parts, one may consider the multiple parts case in which finding the sequence of parts to be processed must also be determined.

Acknowledgement

The authors would like to thank Professor Gerd Finke of Laboratory Leibniz, IMAG, France, and the anonymous reviewers for their helpful comments and suggestions.

Appendix A

Here we will derive the cycle time for the proposed cycle. Assume the robot is waiting idle at the input buffer at time 0. For $i = 1, \ldots, m$, let T_i^{load} represent the time immediately after loading machine i and T_i^{unload} the time the robot arrives at machine i for unloading. We set $D_i = T_i^{\text{unload}} - T_i^{\text{load}}$. Moreover, let w_i be the waiting time of the robot in front of machine i, i.e. $w_i = \max\{0, P - D_i\}$. With our notation, set

$$T_1^{\text{load}} = 2\epsilon + \delta,$$

since the robot takes a part from the input buffer (ϵ), transports it to the first machine (δ) and loads this machine (ϵ). After the robot loads the $(i-1)$th machine, it moves to the input buffer ($(i-1)\delta$), takes a part (ϵ), transports it to the ith machine ($i\delta$) and loads this machine (ϵ). In other words,

$$T_i^{\text{load}} = T_{i-1}^{\text{load}} + (2i-1)\delta + 2\epsilon, \quad \text{for } i = 2, \ldots, m.$$

Before arriving at the first machine for unloading, the robot loads the mth machine and moves to the first machine, i.e.

$$T_1^{\text{unload}} = T_m^{\text{load}} + (m-1)\delta = 2m\epsilon + (m^2 + m - 1)\delta.$$

Before unloading machine i, the robot has to first unload machine $i-1$ ($T_{i-1}^{\text{unload}} + w_{i-1} + \epsilon$), drop the part to the output buffer ($(m - i + 2)\delta + \epsilon$) and return to machine i ($(m - i + 1)\delta$). Hence, for $i = 2, \ldots, m$,

$$T_i^{\text{unload}} = T_{i-1}^{\text{unload}} + (2m - 2i + 3)\delta + 2\epsilon + w_{i-1}.$$

Now, using the above relationships it is easy to obtain

$$D_1 = 2(m-1)\epsilon + (m^2+m-2)\delta \quad \text{and} \quad D_i = D_{i-1} + (2m-4i+4)\delta + w_{i-1},$$

and

$$D_i = D_1 + 2(i-1)(m-1)\delta + w_1 + \cdots w_{i-1}, \quad \text{for } i = 2,\ldots,m.$$

Now, if $w_1 > 0$, in other words $w_1 = P - D_1$, then $D_i \geq P$ for $i = 2,\ldots,m$ and, therefore, $w_2 = w_3 = \cdots w_m = 0$. If $w_1 = 0$, that is to say $P \leq D_1$, then also $P \leq D_i$ for $i = 2,\ldots,m$, since $D_1 \leq D_i$ and again we have $w_2 = w_3 = \cdots = w_m = 0$. The total time to produce m parts with the proposed cycle is

$$T_m^{\text{unload}} + \epsilon + \delta + \epsilon + (m+1)\delta,$$

and after substituting for the easily calculated value of T_m^{unload}, this becomes

$$4m\epsilon + 2m(m+1)\delta + \max\{0, P - 2(m-1)\epsilon - (m-1)(m+2)\delta\}.$$

Consequently, the cycle time of the proposed cycle with m machines is

$$T_{\text{proposed}(m)} = 4\epsilon + 2(m+1)\delta + \frac{1}{m}(\max\{0, P - 2(m-1)\epsilon - (m-1)(m+2)\delta\}). \quad \text{(A1)}$$

Appendix B

Cycle	Activity sequence	Cycle time
$C1$	$A_{01}A_{02}A_{13}A_{23}$	$4\epsilon + 6\delta + 1/2(\max\{0, P - 2\epsilon - 4\delta\})$
$C2$	$A_{01}A_{02}A_{23}A_{13}$	$4\epsilon + 6\delta + P/2$
$C3$	$A_{01}A_{13}A_{02}A_{23}$	$4\epsilon + 6\delta + P$
$C4$	$A_{01}A_{13}A_{23}A_{02}$	$4\epsilon + 6\delta + P/2$
$C5$	$A_{01}A_{23}A_{13}A_{02}$	$4\epsilon + 7\delta + 1/2(\max\{0, P - 2\epsilon - 4\delta\})$
$C6$	$A_{01}A_{23}A_{02}A_{13}$	$4\epsilon + 7\delta + 1/2(\max\{0, P - 4\epsilon - 8\delta\})$

References

Akturk, M.S., Gultekin, H. and Karasan, O.E., Robotic cell scheduling with operational flexibility. *Discr. Appl. Math.*, 2005, **145**, 334–348.

Brauner, N. and Finke, G., On the conjecture in robotic cells: new simplified proof for the three-machine case. *INFOR*, 1999, **37**, 20–36.

Browne, J., Harhen, J. and Shivnan, J., *Production Management Systems*, 1996 (Addison-Wesley: New York).

Crama, Y. and Van de Klundert, J., Cyclic scheduling of identical parts in a robotic cell. *Oper. Res.*, 1997, **45**, 952–965.

Crama, Y. and Van de Klundert, J., Cyclic scheduling in 3-machine robotic flow shops. *J. Schedul.*, 1999, **4**, 35–54.

Crama, Y., Kats, V., van de Klundert, J. and Levner, E., Cyclic scheduling in robotic flowshops. *Ann. Oper. Res.*, 2000, **96**, 97–124.

Geismar, H.N., Dawande, M. and Sriskandarajah, C., Approximation algorithms for k-unit cyclic solutions in robotic cells. *Eur. J. Oper. Res.*, 2005, **162**, 291–309.

Gultekin, H., Akturk, M.S. and Karasan, O.E., Scheduling in a three-machine robotic flexible manufacturing cell. *Comput. Oper. Res.*, 2007, **34**, 2463–2477.

Hall, N.G., Kamoun, H. and Sriskandarajah, C., Scheduling in robotic cells: classification, two and three machine cells. *Oper. Res.*, 1997, **45**, 421–439.

Hall, N.G., Kamoun, H. and Sriskandarajah, C., Scheduling in robotic cells: complexity and steady state analysis. *Eur. J. Oper. Res.*, 1998, **109**, 43–65.

Hall, N.G., Potts, C.N. and Sriskandarajah, C., Parallel machine scheduling with a common server. *Discr. Appl. Math.*, 2000, **102**, 223–243.

Han, B.T. and Cook, J.S., An efficient heuristic for robot acquisition and cell formation. *Ann. Oper. Res.*, 1998, **77**, 229–252.

Logendran, R. and Sriskandarajah, C., Sequencing of robot activities and parts in two-machine robotic cells. *Int. J. Prod. Res.*, 1996, **34**, 3447–3463.

Mata, V. and Tubaileh, A., The machine layout in robot cells. *Int. J. Prod. Res.*, 1998, **36**, 1273–1292.

Sethi, S.P., Sriskandarajah, C., Sorger, G., Blazewicz, J. and Kubiak, W., Sequencing of parts and robot moves in a robotic cell. *Int. J. Flex. Mfg Syst.*, 1992, **4**, 331–358.

Sriskandarajah, C., Hall, N.G. and Kamoun, H., Scheduling large robotic cells without buffers. *Ann. Oper. Res.*, 1998, **76**, 287–321.