Good Production Cycles for Circular Robotic Cells

Florence Thiard, Nicolas Catusse, Nadia Brauner

Good Production Cycles for Circular Robotic Cells

Florence Thiard, Nicolas Catusse, Nadia Brauner

Univ. Grenoble Alpes, G-SCOP, F-38000 Grenoble, France
CNRS, G-SCOP, F-38000 Grenoble, France
e-mail: {florence.thiard, nicolas.catusse, nadia.brauner}@g-scop.grenoble-inp.fr

October 30, 2015

Abstract

In this paper, we study cyclic production for throughput optimization in robotic flow-shops. We are focusing on simple production cycles. Robotic cells can have a linear or a circular layout: most classical results on linear cells cannot be extended to circular cells, making it difficult to quantify the potential gain brought by the latter configuration. Moreover, though the problem of finding the best one part production cycle is polynomial for linear cells, it is NP-hard for circular cells.

We consider the special case of circular balanced cells. We first consider three basic production cycles, and focus on one which is specific to circular cells, for which we establish the expression of the cycle time. Then, we provide a counter-example to a classical conjecture still open in this configuration. Finally, based on computational experiments, we make a conjecture on the dominance of a family of cycle, which could lead to a polynomial algorithm for finding the best 1-cycle for circular balanced cells.

1 Positioning of the problem

Robotic cells consist in a flowshop where the machines are served by a robot. Present in many industries, they are frequently used in semi-conductor manufacturing and electroplating (Dawande et al., 2007). The model was first introduced by Asfahl (1985) to describe a production cell for truck differentials, while Sethi et al. (1992) provided the first formal study for small dimension cells. The robotic flow-shop problem for the production of multiple part types has been proven NP-complete for 3 machines by Hall et al. (1998).

As robotic cells constitute an adequate environment for large-scale production of a few different types of products, the throughput – the number of part produced per time unit – is a natural measure for their performance.Dawande et al. (2007) present a survey on throughput optimization in robotic cells.

1.1 Notations and problem specification

Formally, a robotic cell consists of m machines, denoted by $M_1, M_2, ..., M_m$, and a robot in charge of the handling of the parts in-between machines. The cell is also equipped with an input buffer,

which provides the parts to be produced in infinite quantity, and an output buffer, also of infinite capacity. These buffers are modeled by two additional machines, respectively M_0 and M_{m+1}. As in a classical flow-shop, all parts must be processed successively on machines $M_1 \dots M_m$ in that order.

The input of the problem consists of travel times, processing times and loading/unloading times. In the general case, travel times and processing times are machine-dependent: $\delta_{i,j}$ denotes the travel time between machines M_i and M_j while $p_{i,j}$ represents the processing time of a part j on machine i. Loading and unloading times are generally assumed identical and denoted by ϵ.

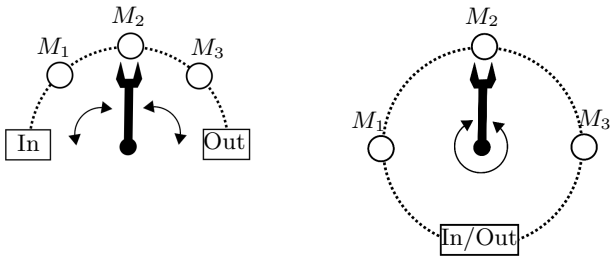
However, if the robot travels at a constant speed, with no acceleration in-between machines, then the only information needed for travel times is the time between two consecutive machines δ_i. In this case, the travel times are called additive: this is a fairly common assumption. Additionally, if the time between any two consecutive machines is the same (if the machines are regularly disposed), the cell is called regular. In cases where the cell is used to group operations of similar length, it is relevant to consider machine-independent processing times. The cell is then called balanced.

Note that for a regular balanced cell producing one type of part, the problem input consists of only 4 numbers, m, δ, p, ϵ.

Depending on the type of robot used, the machines and the input and output stations can be disposed in several ways. Two main configurations are studied in the literature: on the one hand, linear or semi-circular layouts (Figure 1(a)), where the input and output buffers are separated and located respectively at each end of the line (Crama and van de Klundert, 1997), and on the other hand, circular layouts (Figure 1(b)), where the machines are arranged in a circle, with the input and output buffers either occupying the same spot ($M_0 = M_{m+1}$), or very close (Rajapakshe et al., 2011; Jung et al., 2015).

This paper focuses on the classical robotic cell model, which means that the cell is served by a single robot which can hold a single part at a time, and the machines are bufferless: as a consequence, a machine cannot be loaded with a new part until processing is finished and the part transferred onto the next machine.

We will also assume that parts may stay on a given machine as long as necessary after processing is finished (this is referred to as unbounded waiting-times or free-pickup criterion). Other policies include no-wait (the part must be retrieved as soon as the processing is finished), and interval (waiting times are bounded), also called Hoist Scheduling Problems (HSP).



(a) semi-circular layout (b) circular layout

Figure 1: Three-machine robotic cells

1.2 Identical parts production

In the general case, where multiple part types must be produced, two types of decisions must be made: sequencing the parts, and scheduling the robot moves. In the case where only one type of part is to be produced, the part sequencing is of course trivial: the problem reduces to finding an optimal robot move schedule. Brauner (2008) presents a survey on single part-type production in robotic cells.

In this case, robot move sequences can be described using the concept of activities, introduced by Crama and van de Klundert (1999). Activities are elementary robot moves defined as such: for $i \in \{0 \dots m\}$, activity A_i refers to the following sequence of events:

1. The robot unloads a part from M_i;
2. The robot travels to M_{i+1};
3. The robot loads the part onto M_{i+1}.

1.3 Cyclic programmation

For large-scale production, it is operationally relevant to prefer a cyclic programmation. This means that the robot repeats indefinitely the same move sequence, each iteration leaving the cell in the same state, with the same machines loaded and the same machines empty. Moreover, cyclic programmations are dominant (Dawande et al., 2005a), which means that for any set of parameters, there always exists an optimal programmation which is cyclic. The elementary sequence is called a cycle.

One-cycles are particular cycles which produce one part exactly: during one iteration, exactly one part enters the cell at M_0, and one processed part leaves the cell. More generally, a k-cycle is a cycle of production of k parts. One-cycles are easy to describe and enumerate using the concept of activities, as they are exactly the permutations of the $m+1$ activities (Crama and van de Klundert, 1999). They are also easier to implement operationally.

As a consequence, it is convenient to restrict the possible move sequences to 1-cycles only. But does this allow to find an optimal sequence? Sethi et al. (1992) formulate the 1-cycle conjecture:

Conjecture 1 (1-cycle conjecture Sethi et al. (1992)) *The set of 1-cycles is dominant (for any set of parameters, there always exists a 1-cycle which is optimal).*

Unfortunately, this conjecture has been proven false on the general case for more than 4 machines (Dawande et al., 2005b; Brauner and Finke, 2001), meaning that 1-cycles are not generally optimal. However, it is interesting to consider their performance compared to general cycles and their *dominance for special cases of robotic cells*, as well as *finding the best 1-cycle*. In this paper, we are interested in the last two problems.

1.4 Impact of the layout

The answers to these questions depend on the cell layout. In fact, although requiring more sophisticated robots, circular layout can improve the travel time, as the robot takes the shortest path around the cell. For example, in a regular balanced cell, travel time between machine M_i and M_j is $\delta_{i,j} = |i-j|\delta$, while on a similar cell with circular layout, it is $\delta_{i,j} = \min(|i-j|, m+1-|i-j|)\delta$.

In order to make a decision regarding the layout of the cell or quantify the potential gain of a circular layout, it is necessary to study the best programmation for either layout. However, dominant sequences for linear cells might not be dominant with a circular layout (Geismar et al., 2005). Most studies on circular cells consider models which relax the blocking constraints one way or another: robot with swapping ability (Jolai et al., 2012), dual-gripper robots (Sethi et al., 2001; Jung et al., 2015; Drobouchevitch et al., 2006), or machine buffers (Drobouchevitch et al., 2010). On the contrary, circular classical single gripper cells, studied by (Geismar et al., 2005; Rajapakshe et al., 2011; Jung et al., 2015) are not as well understood yet as their linear counterparts.

1.4.1 1-cycle conjecture

The 1-cycle conjecture is valid for 2-machine cells regardless of the layout (Sethi et al., 1992). For linear layouts, it is valid for 3-machine cells (Crama and van de Klundert, 1997), and false for 4-machine cells (Brauner and Finke, 2001). For the special case of regular balanced cell, it is valid up to 15 machines (Brauner, 2008). However, for circular cells with more than 2 machines, it is still open, even for the regular balanced case.

In Section 3 we provide a counter-example to the 1-cycle conjecture for 6-machine regular balanced cells.

1.4.2 Best 1-cycle problem

Finding the best 1-cycle is polynomial in linear additive cells: Crama and van de Klundert (1997) proved the dominance of a family of permutations within 1-cycles, and derived a polynomial algorithm for solving this problem. However, these results do not stand for circular layouts, and Rajapakshe et al. (2011) showed that in a circular regular cell, this same problem is NP-hard. The authors also provide regions of optimality for classical cycles and a $\frac{5}{3}$-approximation of the best 1-cycle. Jung et al. (2015) extend these results to k-cycles.

For the special case of regular balanced cell, the complexity of finding the best 1-cycle is still unknown (Table 1). In Section 4, based on computer simulation, we conjecture that in this case, the problem is polynomial.

Table 1: Complexity of the best 1-cycle problem

	linear layout	circular layout
additive	P (Crama and van de Klundert, 1997)	NP-hard (Rajapakshe et al., 2011)
regular balanced	P (Crama and van de Klundert, 1997)	?

2 Cyclic analysis of circular regular balanced cells

From now on, we consider only regular balanced cells with circular layout. To simplify algebraic expressions, we will assume that loading/unloading times are negligible. Thus, the parameters of the problem are m the number of machine, p the processing times and δ the travel time between two consecutive machines.

We will use c to represent a generic cycle and π a generic 1-cycle.

For any cycle c, we denote the long run average cycle time by $T(c)$. Finding a robot move sequence which minimizes the throughput is equivalent to finding a cycle which minimizes the cycle time divided by the number of parts produced in one iteration (formally, $\frac{T(c_k)}{k}$ for the k-cycle c_k).

In this section we present some classical bounds on the cycle time, and study the region of optimality of three basic cycles. Two of them are already well known for linear configuration; the third one is specific to circular layouts; we give a formulation of its cycle time.

2.1 Bounds on the cycle time

First, we present two classical lower bounds, valid both for linear and circular layout. The formulation is adapted to the regular balanced case.

Proposition 1 *(Crama and van de Klundert, 1997)*
Any k-cycle c verifies

$$T(c) \geq k(p + 4\delta) \tag{1}$$

This bound is the minimum time between two loadings of the same machine.

Proposition 2 *(Dawande et al., 2002)*
Any k-cycle c verifies

$$T(c) \geq k((m+1)\delta + m\min(p, \delta)) \tag{2}$$

Intuitively, if an activity A_i is immediately followed by the subsequent activity A_{i+1}, then the robot waits p time units; if not, it adds at least δ to its minimum travel time.

2.2 Three basic cycles

We call identity cycle (also named uphill permutation or forward cycle in the literature) the cycle $\pi_{id} = (A_0 A_1 ... A_m)$. Trivially, the cycle time of this cycle is

$$T(\pi_{id}) = (m+1)\delta + mp \tag{3}$$

In this cycle, the robot circles the cell once. Intuitively, this cycle is interesting for instances for which p is much smaller than δ. From the lower bound in Proposition 2, one can immediately derive the following:

Proposition 3 *If $p \leq \delta$, then the identity cycle π_{id} is optimal.*

Downhill cycle

We call downhill cycle (also named reverse cycle) the cycle $\pi_d = (A_0 A_m A_{m-1} ... A_1)$. The cycle time of this cycle is

$$T(\pi_d) = 3(m+1)\delta + \max(0, p - (3m-1)\delta) \tag{4}$$

In this cycle, each inter-machine spot is visited by the robot three times. Intuitively, this cycle is interesting for instances for which p is much greater than δ. From the lower bound in Proposition 1, one can easily derive the following:

Proposition 4 *If $p \geq (3m-1)\delta$, then the downhill cycle π_d is optimal.*

Odd-Even cycle

In circular cells, a third 1-cycle of particular interest is the odd-even cycle, defined as such:

For m even,

$$\pi_{oe} = (A_0 A_2 A_4 \dots A_m A_1 A_3 A_5 \dots A_{m-1})$$

and for m odd,

$$\pi_{oe} = (A_0 A_2 A_4 \dots A_{m-1} A_1 A_3 A_5 \dots A_m)$$

In this cycle, the robot circles around the cell twice: the first time performing even activities (thus loading odd machines), the second time performing odd activities (thus loading even machines). This cycle is dominated in cells with linear layout (Crama and van de Klundert, 1997).

2.3 The odd-even cycle

For cells with circular layouts, the odd-even cycle offers a good compromise between travel time and waiting time. The formal expression of its cycle time is not as trivial as the other two basic cycles presented in Section 2.2. Here, we establish this expression for regular balanced cell and give the main ideas of the proof.

Proposition 5 *The cycle time for the odd-even cycle is*

$$T(\pi_{oe}) = 2(m+1)\delta + \frac{2\alpha - 1}{\alpha} \max(0, p - (m+1)\delta)$$

$$\text{with } m = \begin{cases} 2\alpha & \text{if } m \text{ even} \\ 2\alpha - 1 & \text{if } m \text{ odd} \end{cases}$$

Proof To understand this expression of the cycle time, it is convenient to consider α consecutive iterations of π_{oe}, and follow the path of one same part in the cell. Figure 2 presents an example for a 4-machine cell; the position of the robot in the cell is represented as a function of time. Dashed lines represent empty robot moves while solid lines represent loaded robot moves.

The robot travels $2(m+1)\alpha\delta$, circling the cell 2α times. During the first robot loop, the part is loaded on machine M_1 with A_0. It is then unloaded during the second loop with A_1. Between the loading and the unloading, the robot travels $(m+1)\delta$ and the part must stay on M_1 at least p, so the robot must wait $\max(0, p-(m+1)\delta)$. Similarly, during any loop, the part is loaded on a machine, then unloaded and taken to the next machine during the next loop, and the robot must wait $\max(0, p-(m+1)\delta)$. If m is odd, the part exits on the last loop. If m is even, it is loaded on M_m during the last loop.

Eventually, for this one part, the robot must wait at least $(2\alpha - 1)\max(0, p-(m+1)\delta)$ over α iterations of the cycle. Adding the waiting times and travel times, we get the expression above as a lower bound of the cycle time.

We prove that this expression is indeed the exact value of the cycle time by exhibiting a fixed point of the waiting times sequence which realizes it; that part is not detailed here. ■

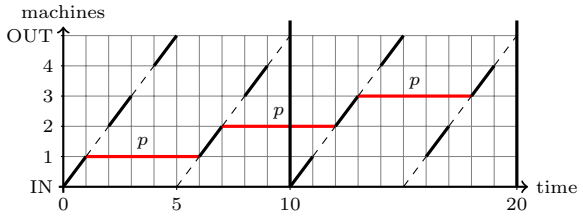


Figure 2: Two consecutive iteration of π_{oe} on a 4-machine cell. If $p \geq 5\delta$, then $T(\pi_{oe}) \geq \frac{1}{2}(3p+5\delta)$.

2.4 Regions of optimality within 1-cycles

We now restrict the study to 1-cycles, and seek to determine the dominance of π_{id}, π_{oe}, π_d over 1-cycles for certain sets of parameters.

For 1-cycles, the bound in Proposition 2 can be slightly improved:

Proposition 6 *If $p \geq \delta$, and π is a 1-cycle with $\pi \neq \pi_{id}$, then*

$$T(\pi) \geq 2(m+1)\delta \tag{5}$$

Based on Propositions 3, 4, 6, and π_{oe} cycle time, we can establish the following region of optimality within 1-cycles:

Proposition 7

(i) if $p \leq \frac{m+1}{m}\delta$, then the identity permutation π_{id} dominates 1-cycles.

(ii) if $\frac{(m+1)}{m}\delta \leq p \leq (m+1)\delta$, then the odd-even cycle π_{oe} dominates 1-cycles.

(iii) if $p \geq 3(m-1)\delta$, the downhill permutation π_d is dominates 1-cycles.

To visualize these regions of optimality, one can use graphic representations such as presented in Figure 3. This graph represents the cycle time as a function of p, for fixed values of δ and m (here, $\delta = 1$, $m = 6$). Note that cycle times are piecewise linear functions in p.

Remain instances where $(m+1)\delta \leq p \leq (3m-1)\delta$, represented by the dashed area on Figure 3.

3 Counter-example to the 1-cycle conjecture

In this section we provide a counter-example to the 1-cycle conjecture for circular cells, and give the main ideas of the proof.

Theorem 1 *In a regular balanced unbounded 6-machine cell with circular layout, the 2-cycle*

$$\hat{c} = (A_0A_2A_5A_4A_1A_6A_0A_3A_2A_5A_1A_4A_3A_6)$$

dominates all 1-cycles for the following instance:

$$\delta = 1 \quad \epsilon = 0 \quad p = 11$$

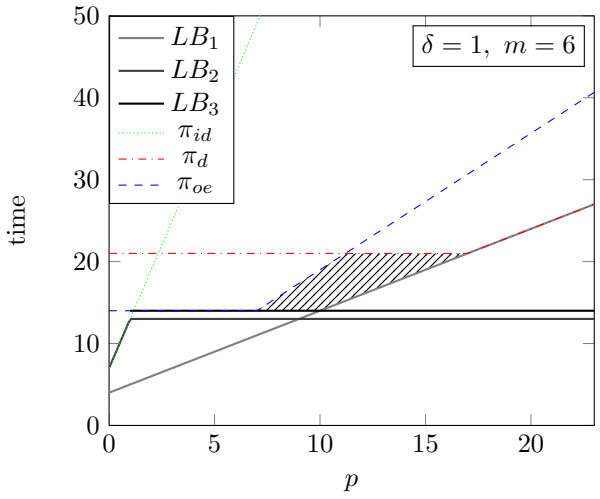


Figure 3: Lower bounds and cycle times of the basic 1-cycles as a function of p. LB_1, LB_2, and LB_3 represent respectively equations (1), (2), and (5)

To prove Theorem 1, we first establish that for 6-machine cells, no 1-cycle performs strictly better than $\{\pi_{id}, \pi_{oe}, \pi_d\}$. For $p \leq (m+1)\delta$ and $p \leq 3m-1$, an optimal cycle is given by Proposition 7. For $(m+1)\delta \leq p \leq (3m-1)\delta$, we establish some necessary structural properties of optimal 1-cycles. Intuitively, such cycles must have their total travel time comprised between two and three times the size of the cell (otherwise they are dominated by either π_{oe} or π_d). They must not contain any sequence of two consecutive activities (otherwise the overall waiting time is too important). Finally, the minimum travel time between the subsequent loading and unloading of a machine must be long enough to avoid excessive waiting at the machine.

On can formalize these conditions and verify that for 6 machines, no 1-cycle satisfies them, therefore the set $\{\pi_{id}, \pi_{oe}, \pi_d\}$ dominates all 1-cycles.

For the parameter set given in Theorem 1, the cycle times of the three basic cycles are:

$$\begin{aligned} T(\pi_{id}) &= 73 \\ T(\pi_{oe}) &= 20 + \frac{2}{3} \\ T(\pi_d) &= 21 \end{aligned}$$

while $\frac{T(\hat{c})}{2} = 20 < T(\pi_{oe})$ (only one time unit of waiting is necessary for each iteration; see Figure 4).

4 Best 1-cycle problem

In this section, we are interested in the problem of finding the best 1-cycle. For a small number of machine ($m \leq 5$), it can be solved by case studies. For $6 \leq m \leq 8$, on can prove that $\{\pi_{id}, \pi_{oe}, \pi_d\}$

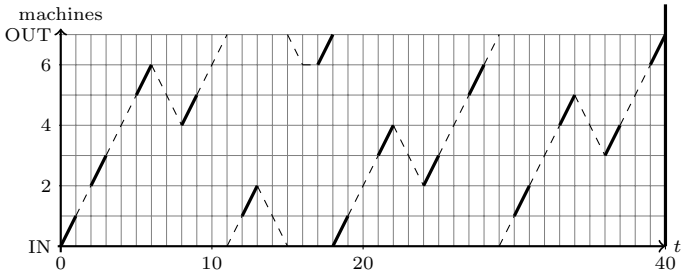


Figure 4: An iteration of $\hat{c}$, for $p = 11$ and $\delta = 1$

dominates all 1-cycles, using the properties establishes in Section 3. For greater values of m, these conditions are not sufficient to filter out all non-optimal 1-cycles.

4.1 Simulation

In order to study the best cycles for larger dimension cells, we developed a tool in Java allowing to simulate the behavior of the cell and therefore experimentally compute cycle times.

The program can be used in two main ways:

- Given an instance of the problem and a list of cycles, compute the cycles times and point out the best ones for this instance. An example is given in Figure 5, for a 10-machine regular balanced cell with $p = 17$ and $\delta = 1$. The most efficient cycles of the list are highlighted: note that for these parameters, the three basic cycles $\{\pi_{id}, \pi_{oe}, \pi_d\}$ (respectively *"cycle identité"*, *"cycle pair-impair"* and *"cycle descendant"*) are dominated.

- Given an instance and the problem and one cycle, produce in LaTeX format the cycle's chronogram (like in Figure 6) over a number of iterations specified by the user. This is useful to identify structural properties of cycle.

Travel times and processing times can be either integer or fractional.

The state of the cell is represented by the position of the robot, and a vector of fractional values representing the state of the machines. If the value for a machine is non-negative, it represents the remaining processing time on this machine. If not, it means the machine is empty.

Brauner (1999) proved that for any cycle, independently of the initial state, the cells reaches a periodic steady state, after a finite number of cycle iterations. This means that there exists a number of iterations (a period) that leaves the cell in exactly the same state, as defined above. Note that if the minimum period is strictly greater than one, cycle time may differ between consecutive iterations. Therefore, to compute the long run average cycle time, it is necessary to consider a full period.

To compute the cycle time, the program first performs several initialization iterations to ensure the cell is running in a periodic steady state (the number of initial iterations can be set by the user). The period is then calculated by monitoring the cell's state after each subsequent iteration. The programs then outputs the average value of the cycle time over one period.

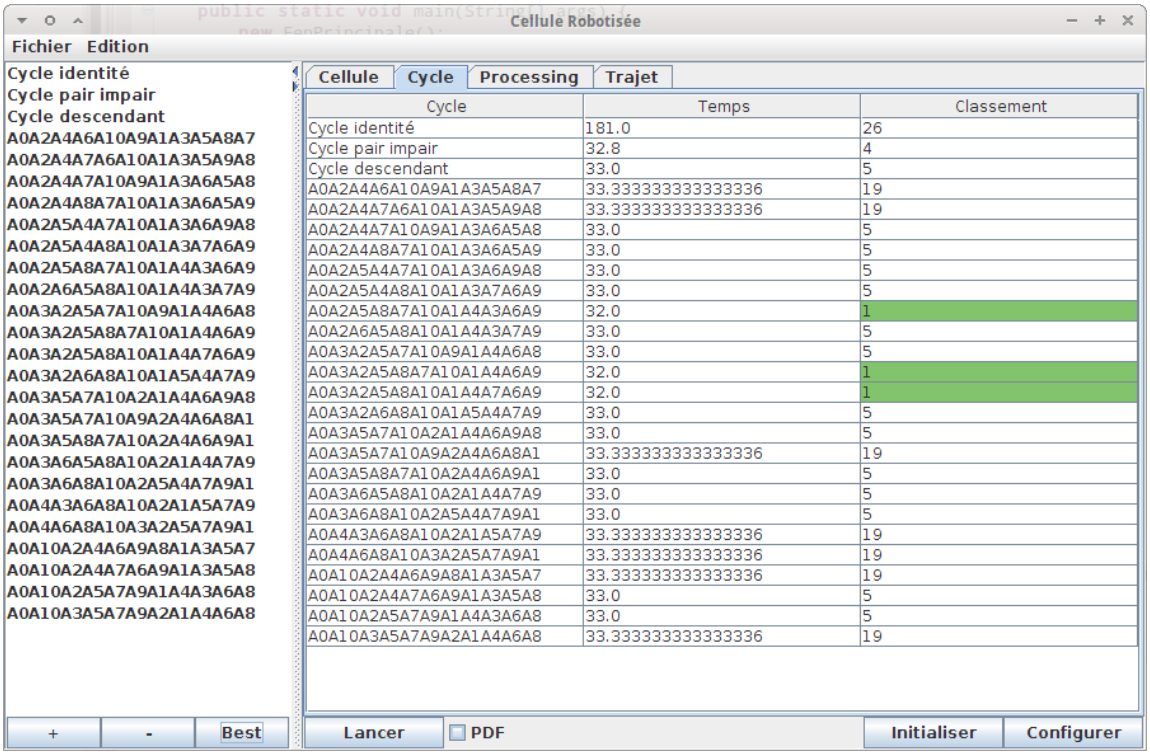


Figure 5: Experimental cycle time computation for a 10-machine cell; the best cycle times are highlighted.

4.2 Experimentations and conjecture

One-cycles are permutations of the $(m+1)$ activities, and as such, their number grows exponentially with the number of machines. Up to 14 machines, it is still possible to generate and test them all.

Table 2 shows the cardinal of a minimum set containing a best 1-cycle for any values of the parameters, up to 14 machines. Although for $m \geq 10$, the set $\{\pi_{id}, \pi_{oe}, \pi_d\}$ is dominated for some values of the parameters, at most 2 additional cycles seem to be necessary to form a dominant set within 1-cycles. This can be formally proved for $m \leq 11$.

Structurally, these additional cycles appear to be similar to the odd-even cycle, but with two slight alterations regularly disposed. For example, for $m = 10$, $\{\pi_{id}, \pi_{oe}, \pi_d, \pi_{2w}\}$ form a dominant set within 1-cycles, with

$$\pi_{2w} = A_0 A_3 A_2 A_5 A_8 A_{10} A_1 A_4 A_7 A_6 A_9$$

represented in Figure 6.

With the odd-even cycle, between a loading and unloading of the same machine, the robot travels exactly $(m+1)\delta$. Intuitively, to reduce waiting times, it is necessary to increase this value, which comes at the price of an increase of the overall travel time. With π_{2w}, the time between a loading and unloading of the same machine is at least $(m+5)\delta$ while the global travel time is only increased by 4δ compared to the odd-even cycle.

Table 2: Cardinal of a minimum dominant set of cycles. Highlighted values ($m \geq 12$) are experimental and not yet confirmed by a formal proof.

m	3	4	5	6	7	8	9	10	11	12	13	14
$\#cycles$	3	4	4	3	3	3	4	4	4	5	4	4

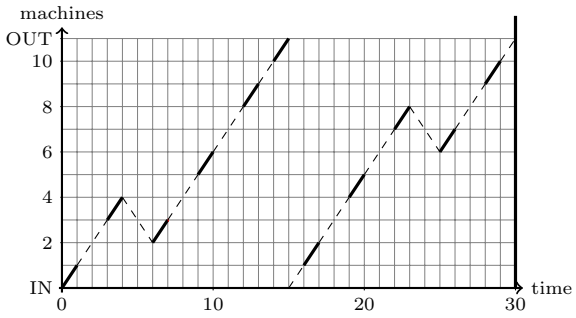


Figure 6: A new dominant cycle for a 10-machine cell: $\pi_{2w} = A_0A_3A_2A_5A_8A_{10}A_1A_4A_7A_6A_9$

The cycle π_{2w} can be defined for any $m \geq 10$, and always dominates $\{\pi_{id}, \pi_{oe}, \pi_d\}$ for some value of the parameters (this can be proved by algebraic formulation of the cycle time).

For m big enough, the definition of π_{2w} can be extended to π_{nw}, formed by similarly altering the odd-even cycle with n regularly disposed alterations.

We conjecture that for any value of $m \geq 10$, a dominant set within 1-cycle can be formed by $\{\pi_{id}, \pi_{oe}, \pi_d, \pi_{2w}\}$ and at most 2 cycles from the family (π_{nw}).

Proving the validity of this conjecture would lead to a polynomial algorithm for finding the best 1-cycle in a regular balanced cell with circular layout.

ACKNOWLEDGMENTS

This research has been partially supported by the LabEx PERSYVAL-Lab (ANR11-LABX-0025).

References

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

Brauner, N. (1999). *Ordonnancement dans des cellules robotisées*. Thèse de doctorat, Université Joseph Fourier, Grenoble, France.

Brauner, N. (2008). Identical part production in cyclic robotic cells: Concepts, overview and open questions. *Discrete Applied Mathematics*, 156(13):2480–2492. SWITZERLAND, AUG, 2006.

Brauner, N. and Finke, G. (2001). Cycles and permutations in robotic cells. *Mathematical and Computer Modelling*, 34(5-6):565–591.

Crama, Y. and van de Klundert, J. (1997). Cyclic scheduling of identical parts in a robotic cell. *Operations Research*, 45(6):952–965.

Crama, Y. and van de Klundert, J. (1999). Cyclic scheduling in 3-machine robotic flow shops. *Journal of Scheduling*, 2:35–54.

Dawande, M., Geismar, H., and Sethi, S. (2005a). Dominance of cyclic solutions and challenges in the scheduling of robotic cells. *SIAM Review*, 47(4):709–721.

Dawande, M., Geismar, H. N., Sethi, S. P., and Sriskandarajah, C. (2005b). Sequencing and scheduling in robotic cells: Recent developments. *Journal of Scheduling*, 8(5):387–426.

Dawande, M., Sriskandarajah, C., and Sethi, S. (2002). On throughput maximization in constant travel-time robotic cells. *Manufacturing and Service Operations Management*, 4(4):296–312.

Dawande, M. W., Geismar, H. N., Sethi, S. P., and Sriskandarajah, C. (2007). *Throughput Optimization in Robotic Cells*. Springer.

Drobouchevitch, I. G., Geismar, H. N., and Sriskandarajah, C. (2010). Throughput optimization in robotic cells with input and output machine buffers: A comparative study of two key models. *European Journal of Operational Research*, 206(3):623 – 633.

Drobouchevitch, I. G., Sethi, S. P., and Sriskandarajah, C. (2006). Scheduling dual gripper robotic cell: One-unit cycles. *European Journal of Operational Research*, 171(2):598 – 631.

Geismar, H., Sethi, S., Sidney, J., and Sriskandarajah, C. (2005). A note on productivity gains in flexible robotic cells. *International Journal of Flexible Manufacturing Systems*, 17(1):5–21.

Hall, N. G., Kamoun, H., and Sriskandarajah, C. (1998). Scheduling in robotic cells: Complexity and steady state analysis. *European Journal of Operational Research*, 109(1):43 – 65.

Jolai, F., Foumani, M., Tavakoli-Moghadam, R., and Fattahi, P. (2012). Cyclic scheduling of a robotic flexible cell with load lock and swap. *Journal of Intelligent Manufacturing*, 23(5):1885–1891.

Jung, K. S., Geismar, H. N., Pinedo, M., and Sriskandarajah, C. (2015). Approximations to optimal sequences in single-gripper and dual-gripper robotic cells with circular layouts. *IIE Transactions*, 47(6):634–652. Accepted manuscript.

Rajapakshe, T., Dawande, M., and Sriskandarajah, C. (2011). Quantifying the impact of layout on productivity: An analysis from robotic-cell manufacturing. *Operations Research*, 59(2):440–454.

Sethi, S., Sidney, J., and Sriskandarajah, C. (2001). Scheduling in dual gripper robotic cells for productivity gains. *IEEE Transactions on Robotics and Automation*, 17(3):423–341.

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