Identical part production in cyclic robotic cells: a state of the art

Nadia Brauner

To cite this version:
Nadia Brauner. Identical part production in cyclic robotic cells: a state of the art. 2005. hal-00082744

HAL Id: hal-00082744
https://hal.archives-ouvertes.fr/hal-00082744
Submitted on 28 Jun 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Identical part production in cyclic robotic cells: 

a state of the art

Nadia Brauner
Identical part production in cyclic robotic cells:

a state of the art

Nadia Brauner

December 29, 2005

Robotic flow-shops consist of \( m \) machines served by a single central robot. They were first introduced by [3] and studied by [33]. In [3], a line for machining castings for truck differential assemblies is described in the form of a 3-machine robotic cell where a robot has to transfer heavy mechanical parts between large machines. The system contains a conveyor belt for the incoming parts and another one for the outgoing parts. In this particular system the robot is not able to traverse the conveyor. Therefore, the movement of the robot from the output to the input station has to traverse the entire cell.

The original application has the form of a flow-shop. However, robotic cells may have very flexible configurations. The robot can easily access the machines thus producing a large variety of products in form of a job-shop. But it is known that the robotic scheduling problem is already NP-hard for a flow-shop with \( m \geq 3 \) machines and two or more different part types [24]. It remains the interesting case of the \( m \)-machine robotic cell in which one wants to produce a single batch of identical parts. We shall mainly concentrate on this case. The robot may have unit capacity, as will be the case in our model, or one may have two-unit robots [35, 34]. Robotic cells with buffers at the machine have been studied in [22, 12]. We concentrate here on the no-buffer case where the parts are either on a machine or transported by the robot. A survey on general robotic cells can be found in [16, 21].

This document gives a state of the art on cyclic scheduling of identical parts in robotic cells. It describes classical configurations of robotic cells (Section 1) and introduces basic concepts and tools for dealing with cyclic productions (Section 2). Then it describes the known results for the classical case where the parts can remain on the machine waiting for the robot (Section 3) and for the no-wait case where the parts have to be removed from the machine as soon as their processing is finished (Section 4). Then a detailed list of open questions is given (Section 5).

1 Robotic cells

The \( m \) machines of a robotic cell are denoted by \( M_1, M_2 \ldots M_m \) and we add two auxiliary machines, \( M_0 \) for the input station IN and \( M_{m+1} \) for the output station OUT (Figure 1).
The raw material for the parts to be produced is available in unlimited quantity at $M_0$. The central robot can handle a single unit at a time. A part is picked up at $M_0$ and transferred in succession to $M_1$, $M_2$ ... $M_m$, where it is machined in this order until it finally reaches the output station $M_{m+1}$. At $M_{m+1}$, the finished parts can be stored in unlimited amounts. We focus on the classical case as in [33], where the machines $M_1$, $M_2$ ... $M_m$ are without buffer facility. In this case, the robot, with unit capacity, has to be empty whenever it wants to pick up a part at $M_h$ ($h = 0, 1 \ldots m$).

![Robotic cell with $m = 4$ machines](image)

Consider an instance $I$ of an $m$-machine robotic cell. Different cell configurations have been studied, depending mainly on the production constraints (Section 1.1) and on the metric for the travel times (Section 1.2).

### 1.1 The production constraints

The processing starts as soon as a part is loaded on a machine. The *processing time* represents the minimum time a part must remain on a machine. If all parts are different, $p_{ij}$ denotes the processing time of part $j$ on machine $M_i$. If one wants to produce one large batch of identical parts, the processing times of the parts on machine $M_i$ are $p_i = p_{ij}$. In the *balanced* case, all processing times are equal, i.e., $p_i = p$ for all $i$.

Once the part is finished, two policies may apply. In the no-wait case, the part must be removed immediately from the machine and transferred to the following machine. In the unbounded case, the part can remain on the machine waiting for the robot.

A classical extension of those two cases is the so-called Hoist Scheduling Problem (HSP) for which the processing policy is different. For the preceding two cases, the processing time is fixed. For the HSP, the processing time is described by an interval, and the no-wait policy applies. This means that the time part $j$ may remain on machine $M_i$ lays in the interval $[p_{ij}, \overline{p_{ij}}]$ (see Table 1). This applies to chemical treatments were the machines correspond to chemical baths. The HSP is NP-hard even for identical parts and very simple (additive) configurations of cells [18]. There exists a wide literature on this problem that we will not develop here (see e.g. [4]).

Denote by $\epsilon$ the time to load a part onto a machine from the robot or to unload a part.
<table>
<thead>
<tr>
<th>Processing</th>
<th>Waiting Policy</th>
<th>Another Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>no-wait</td>
<td>$p_i$</td>
<td>0</td>
</tr>
<tr>
<td>HSP</td>
<td>in $[p_{ij}, \overline{p}_{ij}]$</td>
<td>0</td>
</tr>
<tr>
<td>unbounded</td>
<td>$p_i$</td>
<td>unbounded</td>
</tr>
</tbody>
</table>

Table 1: Two different expressions of the production constraints

from a machine onto the robot.

1.2 The travel metric of the robot

We shall consider different classical metrics for the travel times of the robot depending on the physical configuration of the cell and on the characteristics of the robot. Denote by $\delta_{h,h'}$, the travel time of the robot (empty or loaded) from $M_h$ to $M_{h'}$. The following natural, and in practice desirable, assumptions are made [13]:

- the travel time from a machine to itself is zero, that is, $\delta_{h,h} = 0$;
- the travel times satisfy the triangle inequality, that is, $\delta_{h,k} + \delta_{k,h'} \geq \delta_{h,h'}$ for all $h$, $k$ and $h'$;
- The travel times are symmetric, that is $\delta_{h,h'} = \delta_{h',h}$ for all $h$ and $h'$.

Travel times verifying those three assumptions are called general (or sometimes “euclidean”, e.g. in [21]). Special configurations of cells have been studied. Table 2 summarizes the most classical ones which we now define in detail.

<table>
<thead>
<tr>
<th>$\delta_{h,h+1}$ not constrained</th>
<th>regular ($\delta_{h,h+1} = \delta$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>general</td>
<td>$\delta_{h,h} = 0$; $\delta_{h,h'} = \delta_{h',h}$; $\delta_{h,h'} \leq \delta_{h,k} + \delta_{k,h'}$</td>
</tr>
<tr>
<td>additive</td>
<td>$\delta_{h,h'} = \delta_{h,k} + \delta_{k,h'}$ for $h &lt; k &lt; h'$</td>
</tr>
<tr>
<td>circular</td>
<td>shortest path along the circle $\delta_{h,h'} = \min(</td>
</tr>
<tr>
<td>constant</td>
<td>$\delta_{h,h'} = \delta$</td>
</tr>
</tbody>
</table>

Table 2: Some classical metrics for the robot travel times

For additive times, to travel between distant machines, the robot passes through all intermediate machines and its speed is constant. This metric is the most popular since, in practice, it is applicable if the machines are on a circle or on a line and if the cell is dense (the robot does not have time to speed up between distant machines). In this case, one has the triangle equality $\delta_{h,h'} = \delta_{h,k} + \delta_{k,h'} = \sum_{k=h}^{h'-1} \delta_{k,k+1}$ for any $h < h'$. By symmetry, this also defines $\delta_{h,h'}$ for $h > h'$. Some authors (e.g. [25, 28]) consider an extension of this case having a constant gain $\gamma$ when travelling between distant machines, i.e., $\delta_{h,h'} = \sum_{k=h}^{h'-1} \delta_{k,k+1} - (h' - h - 1)\gamma$ for $h < h'$. We assume $\gamma = 0$.  

3
In circular cells, the input and the output stations coincide, i.e. \( M_0 = M_{m+1} \). Thus the robot chooses the shortest path along the circle formed by the machines. Then, the travel times verify \( \delta_{h,h'} = \min(\sum_{k=h}^{h-1} \delta_{k,k+1} + \sum_{k=h}^{h'} \delta_{k,k+1} + \sum_{k=0}^{h-1} \delta_{k,k+1}) \) for any \( h < h' \).

In the regular case, the machines are equidistant and we denote \( \delta_{h,h'+1} = \delta \). This constraint can be added to additive (as in the seminal paper [33]) or to circular cells.

For constant travel times (introduced in [20]), \( \delta \) is the time for the robot to travel between any two distinct machines \( M_h \) and \( M_{h'} \): \( \delta_{h,h'} = \delta \). The interest of this metric is that it is simpler to study than the others but it seems to have the same properties as the general additive metric.

## 2 Activities and \( k \)-cycles

The robotic scheduling problem is already NP-hard for a flow-shop with \( m \geq 3 \) machines and two or more different part types [24]. Therefore, we concentrate on the interesting case of the \( m \)-machine robotic cell in which one wants to produce identical parts. Then the problem reduces to finding the optimal strategy for the robot moves in order to obtain the maximal throughput rate for this unique part.

In [21], the authors prove that there always exists a cyclic production that is optimal. Therefore, we consider cyclic robot moves for the production process of the parts and define a \( k \)-cycle as a production cycle of exactly \( k \) parts. It can be described as a sequence of robot moves where exactly \( k \) parts enter the system at \( M_0 \), \( k \) parts leave the system at \( M_{m+1} \) and each time the robot executes the \( k \)-cycle, the system returns to the same state, i.e. the same machines are loaded, the same machines are empty and the robot returns to the starting position. To describe \( k \)-cycles we use the concept of activities [19].

The activity \( A_h \) (\( h = 0, 1 \ldots m \)) consists of the following sequence:
- the idle robot takes a part from \( M_h \);
- the robot travels with this part from \( M_h \) to \( M_{h+1} \);
- the robot loads this part onto \( M_{h+1} \).

Note, that many sequences are not feasible, e.g. \( (\ldots A_0, A_0 \ldots) \), since the robot carries a part to \( M_1 \) which is occupied. In [19], the authors characterize the \( k \)-cycles as follows: A \( k \)-cycle \( C_k \) is a sequence of activities, in which each activity occurs exactly \( k \) times and between two consecutive (in a cyclic sense) occurrences of \( A_h \) (\( h = 1, 2 \ldots m - 1 \)) there is exactly one occurrence of \( A_{h-1} \) and exactly one occurrence of \( A_{h+1} \).

We represent a \( k \)-cycle \( C_k \) as in Figure 2. The horizontal axis represents the time. The vertical axis represents the cell. The graph indicates the position of the robot in the cell while executing the cycle. The dashed lines are the empty robot moves and the plain lines are the loaded robot moves, the loading and unloading processes or the waiting times of the robot at the machines. Let us illustrate this with an example. In a 3-machine regular additive cell, consider the 1-cycle \( C = (A_0 A_2 A_1 A_3) \). Let \( I \) be the following instance:

\[
\delta = 1; \quad \epsilon = 0; \quad p_1 = 6; \quad p_2 = 9; \quad p_3 = 6.
\]

At the beginning of the cycle \( C \), machine \( M_2 \) is loaded and machines \( M_1 \) and \( M_3 \) are
empty. We suppose that, at time 0, the part has been on $M_2$ for 6 time units. At time 9, on Figure 2, the robot is at machine $M_3$ and is waiting one time unit for the part to be ready in order to execute activity $A_3$. One can observe that the cycle does not repeat identically. In this example, if the parts was on $M_2$ for 5.5 time units (instead of 6), the cycle would have repeated identically. However, the mean cycle time (14.5 in both case) does not seem to depend on the initial state (ergodicity). Let $T(C_k)$ be the long run average execution time of the $k$-cycle $C_k$.

Figure 2: Representation of the 1-cycle $C = (A_0, A_2, A_1, A_3)$ for the instance $I$

We call $T(C_k)$ the cycle time and $T(C_k)/k$ the cycle length. The throughput rate is defined by $k/T(C_k)$. Thus the $\rho$-cycle $C_\rho$ is optimal if it maximizes the throughput rate or equivalently minimizes the cycle length $T(C_k)/k$ over the set of all possible $k$-cycles ($k = 1, 2, 3, \ldots$). A set of cycles $S$ is said to be dominant if, for any instance, there exists a cycle of $S$ that is optimal.

2.1 Dominant sets of cycles

Ideally, one would like to determine, for a given instance, an optimal $k$-cycle. However, this is so far not possible, except for very particular cases, for instance for very slow or for very fast robots compared to the processing times. In [33] the authors proposed the following conjecture:

1-cycle Conjecture [33]: The set of 1-cycles is dominant.

This conjecture is valid for 2-machine cells (see Section 2.4) and unbounded 3-machine cells [19, 8]. However it is false for no-wait 3-machine cells [1] and unbounded regular additive or constant 4-machine cells [10, 21]. It has been replaced by the following conjecture:

Agnetis' Conjecture [1]: The set of $k$-cycles with $k \leq m - 1$ is dominant.

Note that this conjecture was originally formulated for additive no-wait cells. Let $S_K$ be the set of all $k$-cycles with $1 \leq k \leq K$. We are interested in the minimal dominant set $S_K$, i.e., $S_K$ is dominant and no $S_{K'}$, is dominant with $K' < K$. We can expect that $K = K(m)$ is a function of the number of machines $m$. We call $K(m)$ the cycle function.

Finding the optimal production cycle can be decomposed into three sub-problems that
we address in the following sections:

(P1) Determine $K$, i.e., find its constant value if it is indeed constant, lower bounds, finite upper bounds...

(P2) Determine the complexity of finding the best cycle in $S_k$ (where $k$ can be any number between 1 and $K$).

(P3) Determine the performance of $S_k$ (where $k$ can be any number between 1 and $K$) defined by

$$P(k) = \max_{\text{instances}} \frac{\text{best cycle length in } S_k}{\text{best cycle length in } S_K}$$

Those three problems have been widely studied especially (P2) and (P3) for $k = 1$ when the 1-cycle Conjecture was still open. In the following sections, we review the state of the art of those problems for the unbounded case (Section 3) and for the no-wait case (Section 4).

2.2 Bounds

In this section, we introduce several bounds on the cycle time. Consider a $k$-cycle $C_k$. We first introduce some inequalities that only depend on the cycle and not on the configuration of the cell. Let $m_h(C_k)$ be the number of times the robot travels between machines $M_h$ and $M_{h+1}$ in both directions during one execution of the $k$-cycle $C_k$ and let $|S|_{C_k}$ be the number of occurrences of the sequence of activities $S$ in $C_k$ and let $u_i(C_k) = |A_i-1 A_i|_{C_k}$. For simplicity, we drop $C_k$ in those notations when no confusion is possible. If the robot never makes any dummy moves, one has [10, 8]:

$$m_0 = 2k$$

$$m_m = 2k$$

$$m_1 = 4k - 2u_1$$

$$m_{m-1} = 4k - 2u_{m-1}$$

$$m_2 \geq 4k - 2u_2 - 2|A_1 A_0 A_2|$$

$$m_{m-2} \geq 4k - 2u_{m-2} - 2|A_{m-2} A_m A_{m-1}|$$

The following inequalities are often used as an optimality criterion. The intuition is that the lower bound is the time elapsed between two successive loadings of machine $M_h$ (see Figure 3). Since this happens $k$ times in a cycle, we have a multiplicative factor $k$.

$$T(C_k) \geq k(\delta_{h,h+1} + \delta_{h+1,h-1} + \delta_{h-1,h} + p_h + 4\epsilon) \quad \forall h = 1, 2 \ldots m$$

(7)

In the additive case it becomes,

$$T(C_k) \geq k(2\delta_{h,h+1} + 2\delta_{h-1,h} + p_h + 4\epsilon) \quad \forall h = 1, 2 \ldots m$$

(8)

and in the constant case one has,

$$T(C_k) \geq k(3\delta + p_h + 4\epsilon) \quad \forall h = 1, 2 \ldots m$$

(9)
One of the interests of the constant case is the fact that the travel time can be expressed easily: one has the following equality for the travel time $T_T(C_k)$ of a $k$-cycle $C_k$:

$$T_T(C_k) = 2k(m + 1)\delta - \sum_{i=1}^{m} u_i \delta$$

(10)

The intuition for this equality is that between two activities, one has a time $\delta$ if and only if the two activities are not consecutive, i.e. they do not participate in a $u_i$. This equality is proved for $k = 1$ in [20].

2.3 State graphs

At each instant in a production cycle, it is possible to calculate the part/machine incidence vector: machine $M_h$ is loaded if and only if the next occurrence of $A_h$ arrives before the next occurrence of $A_{h-1}$. For instance, at the beginning of the execution of the cycle $(A_0, A_4, A_6, A_7, A_5, A_3, A_2, A_1)$, the part/machine incidence vector is $(0, 1, 1, 1, 0, 1, 0)$.

In the state graph $G_m$ associated with an $m$-machine robotic cell, each vertex is a part/machine incidence vector that represents the state of the cell. Therefore, $G_m$ has $2^m$ vertices. The arcs represent the activities of the robot to pass from a state to another state. The state graph $G_3$ is given in Figure 4. In $G_3$, ‘100’ represents the state of the system with machine $M_1$ loaded and machines $M_2$ and $M_3$ empty. To go from state ‘100’ to state ‘010’, the robot transfers a part from $M_1$ to $M_2$, executing activity $A_1$. Each $k$-cycle corresponds uniquely to a cycle of length $k(m + 1)$ in the graph. For instance, the 1-cycle $(A_0, A_3, A_1, A_2)$ corresponds to the sequence of vertices ‘001’, ‘101’, ‘100’, ‘010’.

Figure 3: Illustration of inequalities (8) and (9)
The state graphs allow to find the number of \( k \)-cycles in an \( m \)-machine cell [14]. The algorithm is based on the traversal of the state graph. A label is attached to each cycle. This label is the inverse of the number of times the cycle appears during the traversal. One then obtains the number of \( k \)-cycles by adding the labels of all the \( k \)-cycles encountered. Table 3 displays the number of \( k \)-cycles in an \( m \)-machine cell. One can remark that this number rapidly grows so that it is impossible to find the best production cycle by enumeration of the cycle times of all \( k \)-cycles.

![Figure 4: The state graph \( G_3 \)](image)

To each graph \( G_m \) (with orientations) is associated a line-graph \( LG_m \) with orientations as follows:

- the vertices of \( LG_m \) correspond to the arcs of \( G_m \);
- \((a,a')\) is an arc of \( LG_m \) if and only if there exists, in \( G_m \), a vertex \( v \) which is the tail of \( a \) and the head of \( a' \) as in Figure 5.

The circuits of \( G_m \) and of \( LG_m \) are equivalent. Therefore, one can work with one or with the other graph. The graph \( LG_3 \) is represented on Figure 6. The arcs of \( LG_m \) can be weighted by the travel times of the robot or by some (unfortunately not all) waiting times. Therefore, a minimum mean circuit length in \( LG_m \) is a lower bound to the optimal...
cycle time.

Figure 6: The line-graph $LG_3$ of $G_3$

Let $G$ be an oriented graph with $n$ vertices and a distance on the arcs. A minimum mean circuit $C$ in $G$ is a circuit which minimizes

\[
\text{sum of the distances of the arcs of } C \div \text{number of vertices in } C
\]

Finding a minimum mean circuit is a problem which can be solved in time polynomial in the number of vertices of the graph (see e.g. [26, 2]). Remark that the graph $LG_m$ has an exponential number of vertices $((m + 3)2^{m-2})$. However, this approach has been used to prove many dominance results described in the following sections.

2.4 Special solved cases

In this section, we consider three very simple solved cases: a very slow or very fast robot (compared to the processing times) and the 2-machine case.

Intuitively, when the travel times are much larger than the processing times, it can be interesting to carry out as few robot moves as possible. In this case, it is optimal to enter a part in the cell and to let the robot transfer this part on all the machines successively, waiting each time for the processing to complete. This is done by the 1-
cycle $\pi_0 = (A_0A_1\ldots A_m)$ also known as the identity cycle. Its cycle time is

$$T(\pi_0) = \sum_{h=0}^{m} \delta_{h,h+1} + \delta_{m+1,0} + 2(m+1)e + \sum_{h=1}^{m} p_h$$

On the other side, when the robot is very fast (compared to the processing times), it is interesting to let it do a lot of moves while other parts are being processed on the machines. In this case, the 1-cycle $\pi_d = (A_0A_mA_{m-1}\ldots A_1)$ also known as the downhill cycle is optimal. Its cycle time is

$$T(\pi_d) = \max\left(\sum_{h=0}^{m} \delta_{h,h+1} + \sum_{h=0}^{m-1} \delta_{h+2,h} + 2(m+1)e + \delta_{1,m};
\max_i (p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i+4e})\right)$$

Notice that if $T(\pi_d)$ is equal to the second term of the max operator, then the lower bound in inequality (7) is attained. Therefore, in this case (large processing times), $\pi_d$ is optimal.

Another extreme problem is the 2-machine case. When the robot transfers a part from $M_1$ to $M_2$, both machines are empty. Therefore, any cycle can be decomposed into a sequence of 1-cycles (and its cycle time is the sum of the cycle times of the sub cycles). Therefore, in this case, one-cycles are optimal (whatever the configuration of the cell is). Moreover, two 1-cycles are possible: the identity and the downhill permutation. Just choose the best one.

## 3 The unbounded case

In the seminal paper [33], the authors consider additive regular robotic cells with unbounded waiting times at the machines and present the 1-cycle Conjecture. This section relates the known results on the unbounded case, first describing the 9-year lifetime of the 1-cycle Conjecture for additive cells (Section 3.1). All results are then extended to the constant case (Section 3.2). The interest in 1-cycles and their simplicity induced research on their performance factor (Section 3.3). We then consider the well understood (but still partially open) regular balanced case for which the input is composed of only 4 numbers (Section 3.4).

On our way, we describe the state of problems (P1), (P2) and (P3) (defined in Section 2.1) for different configurations of unbounded robotic cells: Table 4 summarizes known values or bounds for the cycle functions $\mathcal{K}(m)$ depending on the cell configurations. Because of their simplicity, 1-cycles have generate a wide amount of results. Table 5 gives the complexity of finding the best 1-cycles and the performance factor of 1-cycles.
<table>
<thead>
<tr>
<th>travel</th>
<th>production</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5 ≤ m ≤ 15</th>
<th>m ≥ 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>general</td>
<td>1</td>
<td>≥ 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>circular</td>
<td>1</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>additive or constant</td>
<td>1</td>
<td>≥ 4*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>additive regular</td>
<td>balanced</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>balanced</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Cycle function, $K(m)$, in the unbounded case (problem (P1) in Section 2.1) (* means "even for the regular case")

<table>
<thead>
<tr>
<th>travel</th>
<th>production</th>
<th>complexity</th>
<th>performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>general</td>
<td>balanced</td>
<td>NP-hard</td>
<td>4</td>
</tr>
<tr>
<td>circular</td>
<td>balanced</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>additive / constant</td>
<td>polynomial</td>
<td>1,5</td>
<td></td>
</tr>
<tr>
<td>additive</td>
<td>balanced</td>
<td>$O(1)$</td>
<td>?</td>
</tr>
<tr>
<td>constant</td>
<td>balanced</td>
<td>$O(1)$</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5: 1-cycle complexity and performance in the unbounded case (problems (P2) and (P3) for $k = 1$ in Section 2.1)

### 3.1 The 1-cycle Conjecture for additive cells

In 1992, Sethi Sriskandarajah, Sorger, Blazewicz and Kubiak [33] claim that the best production cycle can be achieved by a one cycle (i.e. $K(m) = 1$ for any $m$) in a regular additive cell (called the 1-cycle Conjecture). In the same paper, the authors state that this conjecture is true for 2-machine cells. In 1997, Hall, Kamoun and Sriskandarajah [25] prove, rather technically, that the 1-cycles dominate the 2-cycles in regular 3-machine cells. In 1999, Crama and van de Klundert [19] extend this result to $k$-cycles proving that the 1-cycle Conjecture is valid for 3-machine additive cells (a shorter proof based on the state graph $G(3)$ and on inequalities (1) to (6) is given in [8]). Moreover, in regular additive 4-machine cells, the 1-cycles again dominate the 2-cycles [5].

In 1997, the interest in 1-cycles is enforced when Crama and van de Klundert [17] prove that the best 1-cycle can be found in polynomial time: in [33], the authors prove that a 1-cycle is completely defined by a permutation of the activities. Hence, the number of 1-cycles is $m!$. Normalize, without loss of generality, the sequences $\pi$ of activities and start with activity $A_0$. The 1-cycles are then of the form $\pi = (A_0, A_{i_1}, A_{i_2}, \ldots A_{i_m})$ where $(i_1, i_2, \ldots i_m)$ is a permutation of $\{1,2,\ldots m\}$. Let us consider the 1-cycles $\pi$ belonging to the set of *pyramidal permutations*. We call $\pi = (A_0, A_{i_1}, A_{i_2}, \ldots A_{i_m})$ pyramidal if there is a $p$ such that $1 \leq i_1 < \ldots < i_p = m$ and $m > i_{p+1} > \ldots > i_m \geq 1$. In [17], the authors prove that the pyramidal permutations dominate the 1-cycles. They give an algorithm of complexity $O(m^3)$ for the determination of the best pyramidal permutation which is therefore also the complexity of finding the best 1-cycle.

Unfortunately, the 1-cycle Conjecture happened to be false for additive regular 4-machine
cells. Indeed, in 2001, Brauner and Finke [10] described a 3-cycle that is strictly better than all 1-cycles for a regular additive 4-machine cell thus proving that $K(4) \geq 3$ and reviving the question of the complexity of finding the best production cycle in robotic cells.

### 3.2 A metric unifying both the additive and the constant cases

All results presented in Section 3.1 are also valid for the constant case. Indeed, in [21], the authors claim that the graphical proof of the validity of the 1-cycle Conjecture for 3-machine additive cells in [8], also works for the constant case. We conjecture that the proof works for general 3-machine cells with travel times verifying

\[ \delta_{12} + \delta_{23} + \delta_{40} \geq \delta_{24} + \delta_{13} + \delta_{02}. \]

For the complexity of finding the best 1-cycle, define the basic cycles constructed as follow [20]:

- partition the activities $A_1, A_2 \ldots A_m$ into two sets $V_1$ and $V_2$;
- construct a sequence $S$ of activities composed of $A_0$ followed by the activities of $V_2$ in decreasing order;
- insert sequentially the activities of $V_1$ in $S$ in increasing order putting $A_i \in V_1$ just after $A_{i-1}$ in $S$.

Let us illustrate this on an example: $m = 8$ and $V_1 = \{A_1, A_2, A_4, A_8\}$ and $V_2 = \{A_3, A_5, A_6, A_7\}$ make the 1-cycle $A_0A_1A_2A_7A_8A_6A_5A_3A_4$. All 1-cycles that can be constructed as described above belong to the set of basic cycles (of cardinality $2^m - m$).

In [20], the authors prove that in a constant robotic cell, basic cycles dominate 1-cycles and that the best basic cycle can be found in polynomial time. Therefore, one has:

**Theorem 1** [17, 20] In additive or constant robotic cells, the best 1-cycle can be found in polynomial time.

In the quest for $K(m)$, the 1-cycle Conjecture was replaced by Agnetis’ Conjecture that claims that $K(m) \leq m - 1$. This conjecture was again proved to be false for 4-machine cells:

**Proposition 1** [11] In a 4-machine cell, the 4-cycle

\[ C_4 = (A_0A_1A_0A_3A_4A_2A_1A_0A_3A_4A_2A_1A_4A_3A_2A_0A_1A_4A_3A_4A_2) \]

strictly dominates all k-cycles for $k = 1, 2, 3$ for the following instance:

- $m = 4$; \quad $\delta = 1$; \quad $\epsilon = 0$; \quad $p_1 = 0$; \quad $p_4 = 0$;
- in the additive case: \quad $p_2 = 10$; \quad $p_3 = 10$;
- in the constant case: \quad $p_2 = 6$; \quad $p_3 = 6$.

We conjecture that Proposition 1 is still valid with

- the following metric that generalizes the additive and the constant case: let $\delta_i$ be the time to travel between two machine with distances in the production process.
equal to $i$: $\delta_{kl} = \delta_{|l-k|}$ with the following assumptions

$$\delta_i \leq \delta_j \text{ for } i \leq j \quad \text{and} \quad 0, 4 < \frac{\delta_2}{\delta_1 + \delta_3} < 1$$

- the same instance generalized to $p_2 = p_3 = 3\delta_1 + 2\delta_2 + \delta_3$.

The preceding extensions of the properties of the additive case to the constant case raise two natural questions:

- Is there a metric which generalizes both cases keeping the known results on the optimality and the complexity of 1-cycles?
- Is there a natural way to extend all proofs for the constant case (which seems simpler to study) to the additive case?

### 3.3 Performance of 1-cycles

Whenever the 1-cycle Conjecture is false, it is interesting to study the quality of 1-cycles since they are simple, easy to implement and the problem reduced to 1-cycles is polynomially solvable for most cases. Let us define the performance factor of 1-cycles, $\mathcal{P}(1)$, as in Section 2.1:

$$\mathcal{P}(1) = \max_{\text{instances}} \frac{\text{cycle length of the best 1-cycle}}{\text{cycle length of the best cycle in } S_{\mathcal{C}}}$$

The 1-cycle $\pi_d$ allows to find the following bounds:

\[
\mathcal{P}(1) \leq \left( 2 - \frac{\delta_0,1 + \delta_{m,m+1}}{\delta_{0,1} + \delta_{m,m+2} + \sum_{i=1}^{m-1} \delta_{i,i+1}} \right) \leq 2 \text{ in the additive case [9, 17]};
\]
\[
\mathcal{P}(1) \leq \left( 2 - \frac{\delta}{m+2} \right) \leq 2 \text{ in the constant case [21]};
\]
\[
\mathcal{P}(1) \leq 4 \text{ in the general case [23]};
\]

For the additive regular case or for the constant case one has, $\mathcal{P}(1) \leq 1.5$ [23]. For a regular additive 4-machine robotic cell, one can give this value with more precision. We know that $\mathcal{P}(1) \geq 16/15$ with the example presented in Proposition 1. With a rather technical study, one may show that $\mathcal{P}(1) \leq 9/8$ and that this number is not tight. Finally, the reduced interval for $\mathcal{P}(1)$ in a regular additive 4-machine cell is given by $[1.06666; 1.125]$.

### 3.4 Regular balanced cells

In this section, we consider balanced cells where an instance is given by four numbers:
- the number of machines, $m$,
- the travel time, $\delta$ (between consecutive machines in the additive case and between any two machines in the constant case),
– the loading/unloading time, $\epsilon$,
– the processing time, $p$.

This case is interesting since it raises complexity problems: the complete description of a production cycle (as a $k$-cycle for instance) is not polynomial in the instance length (see [7] for a discussion on this topic). We shall discuss the complexity of finding the best 1-cycle and the optimality of 1-cycles for the constant (Section 3.4.1) and the additive (Section 3.4.2) cases.

### 3.4.1 Constant

For the constant balanced case, the problem is easy. We prove that, if $p \leq \delta$ then the identity cycle $\pi_0 = (A_0A_1 \ldots A_m)$ is optimal, otherwise, the downhill permutation $\pi_d = (A_0A_mA_{m-1} \ldots A_1)$ is optimal.

**Theorem 2** In the constant balanced case, one has $K(m) = 1$ and the best 1-cycle can be found in constant time.

**Proof** Since each $u_i = |A_{i-1}A_i|$ generates a waiting time of $p$, inequality (10) indicates that the cycle time of any $k$-cycle $C_k$ satisfies

$$T(C_k) \geq 2k(m + 1)\delta + \sum_{i=1}^{m} u_i(p - \delta) + 2k(m + 1)\epsilon$$

Moreover, the cycle times of the identity cycle and of the downhill permutation are

$$T(\pi_0) = (m + 2)\delta + mp + 2(m + 1)\epsilon$$
$$T(\pi_d) = 2(m + 1)\delta + 2(m + 1)\epsilon + \max(0, p - (2m - 1)\delta - 2(m - 1)\epsilon)$$

If $p \geq (2m - 1)\delta + 2(m - 1)\epsilon$, then $\pi_d$ achieves the lower bound given in inequality (9) and hence, $\pi_d$ is optimal. For the following, we shall assume that $p \leq (2m - 1)\delta + 2(m - 1)\epsilon$ so that $T(\pi_d) = 2(m + 1)\delta + 2(m + 1)\epsilon$.

If $p \geq \delta$ then $T(C_k) \geq 2k(m + 1)\delta + 2k(m + 1)\epsilon = kT(\pi_d)$ and $\pi_d$ is at least as good as $C_k$. If $p \leq \delta$, since $u_i \leq k$ and $p - \delta \leq 0$, one has

$$T(C_k) \geq 2k(m + 1)\delta + mk(p - \delta) + 2k(m + 1)\epsilon = k(m + 2)\delta + 2k(m + 1)\epsilon + mkp = kT(\pi_0)$$

which proves that $K(m) = 1$. A polynomial time algorithm for the best 1-cycle would return $\pi_0$ if $p \leq \delta$ and $\pi_d$ otherwise.

### 3.4.2 Additive regular case

In additive regular balanced cells, the best 1-cycle can also be found in polynomial time [13]. We only consider the case $m \geq 4$, since for $m \leq 3$, for the more general additive
case, the 1-cycles are dominant and finding the best 1-cycle can be done in polynomial time.

Consider the $\frac{m}{2} + 1$ following 1-cycles for $m \geq 4$ and $m$ even:

$$
\begin{align*}
\pi_0 &= (A_0 A_1 A_2 \ldots A_m) \\
\pi_\alpha &= (A_0 A_{\alpha+1} A_{\alpha+3} \ldots A_{m-\alpha-1} A_m A_{m-1} A_{m-2} \ldots A_{m-\alpha} A_{m-\alpha-2} ) \\
& \quad A_{m-\alpha-4} \ldots A_{\alpha+2} A_{\alpha-1} A_{\alpha-2} \ldots A_2 A_1) \text{ for } \alpha \in [1, \frac{m}{2} - 1] \\
\pi_{m/2} &= (A_0 A_m A_{m-1} \ldots A_2 A_1) \pi_d
\end{align*}
$$

Theorem 3 [13] In an additive regular balanced cell, if $m$ is even, the best 1-cycle is

$$
\begin{align*}
\pi_0 &\text{ if } 0 \leq p \leq \delta \\
\pi_\alpha &\text{ if } (4\alpha - 3)\delta + 2(\alpha - 1)\epsilon \leq p \leq (4\alpha + 1)\delta + 2\alpha \epsilon \quad \alpha \in [1, \frac{m}{2} - 1] \\
\pi_{m/2} &\text{ if } (2m - 3)\delta + (m - 2)\epsilon \leq p.
\end{align*}
$$

The case with $m$ odd is similar. Consider the following 1-cycles for $m \geq 5$ and $m$ odd:

$$
\begin{align*}
\pi_0 &= (A_0 A_1 A_2 A_3 \ldots A_m) \\
\pi_1^0 &= (A_0 A_1 A_3 \ldots A_{m-2} A_m A_{m-1} A_{m-3} \ldots A_4 A_2) \\
\pi_0^1 &= (A_0 A_2 A_4 \ldots A_{m-1} A_m A_{m-2} A_{m-4} \ldots A_3 A_1) \\
\pi_1^\alpha &= (A_0 A_{\alpha+1} A_{\alpha+3} \ldots A_{m-\alpha-2} A_m A_{m-1} A_{m-2} \ldots A_{m-\alpha-1} \\
& \quad A_{m-\alpha-3} A_{m-\alpha-5} \ldots A_{\alpha+2} A_{\alpha-1} A_{\alpha-2} \ldots A_2 A_1) \\
\pi_2^\alpha &= (A_0 A_{\alpha+2} A_{\alpha+4} \ldots A_{m-\alpha-1} A_m A_{m-1} A_{m-2} \ldots A_{m-\alpha} \\
& \quad A_{m-\alpha-2} A_{m-\alpha-4} \ldots A_{\alpha+3} A_{\alpha-1} A_{\alpha-2} \ldots A_2 A_1) \\
\pi_{m/2}^\alpha &= (A_0 A_m A_{m-1} \ldots A_2 A_1) = \pi_d
\end{align*}
$$

The permutations $\pi_1^\alpha$ and $\pi_2^\alpha$ have the same cycle time.

Theorem 4 [13] In an additive regular balanced cell, if $m$ is odd, the best 1-cycle is

$$
\begin{align*}
\pi_0 &\text{ if } 0 \leq p \leq \delta \\
\pi_1^0 &\text{ if } \delta \leq p \leq 2\delta \\
\pi_1^\alpha &\text{ if } (4\alpha - 2)\delta + 2(\alpha - 1)\epsilon \leq p \leq (4\alpha + 2)\delta + 2\alpha \epsilon \quad \alpha \in [1, \frac{m-3}{2}] \\
\pi_{m/2}^\alpha &\text{ if } (2m - 4)\delta + (m - 3)\epsilon \leq p.
\end{align*}
$$

Therefore, for a given instance, one just has to determine $\alpha$ from $m$, $p$, $\delta$, and $\epsilon$ to obtain the best 1-cycle. However, for additive regular balanced cells, the exact value of $\mathcal{K}(m)$ is not completely known: for $m \leq 3$, for the more general additive cells one has $\mathcal{K}(m) = 1$. For regular additive balanced cells, the detailed proof that 1-cycles are dominant can be found in [5] for $m = 4$ and [6] for $m \leq 6$. This last proof can be extended to $m \leq 15$ with a case by case study. However, the value of $\mathcal{K}(m)$ with $m \geq 16$ is still unknown.
4 The no-wait case

In this section, we review the no-wait case. Agnetis’ conjecture, which claims that $K(m) = m - 1$, was first formulated for this case. Table 6 summarizes known results on the value of $K(m)$ and on the complexity of finding the best production cycle for different values of $m$.

<table>
<thead>
<tr>
<th>$m$</th>
<th>config.</th>
<th>$K(m)$</th>
<th>complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>add. reg.</td>
<td>1</td>
<td>$P$ [1]</td>
</tr>
<tr>
<td>3</td>
<td>add. reg.</td>
<td>2</td>
<td>$P$ [1]</td>
</tr>
<tr>
<td>4</td>
<td>add. reg.</td>
<td>$\geq 3$</td>
<td>open</td>
</tr>
<tr>
<td>4</td>
<td>reg. bal.</td>
<td>3</td>
<td>$P$ [31]</td>
</tr>
<tr>
<td>$\geq 5$</td>
<td>reg. bal.</td>
<td>$\geq m - 1$</td>
<td>open (1-cycles in pol. time but not optimal)</td>
</tr>
</tbody>
</table>

Table 6: Dominance and complexity results for the no-wait additive case

The no-wait case has historically been studied for additive regular cells. For $m \leq 3$, Agnetis’ Conjecture was proved in [1]. For $m = 4$, it has been proved for the balanced case (equal processing times on all machines) in [31]. The idea of the proof is to use the state graph $G_4$ and its line graph. The no-wait constraints allow, for each instance, to erase some forbidden arcs in the graph. In the reduced graph, structural properties allow to prove the desired result. Table 7 describes the optimal cycle for a regular balanced 4-machine cell. The first column gives the corresponding interval for the processing time $p$.

<table>
<thead>
<tr>
<th>$p$</th>
<th>optimal cycle</th>
<th>degree</th>
<th>cycle length</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[0, 4\delta]$</td>
<td>$A_0A_1A_2A_3A_4$</td>
<td>1</td>
<td>$4p + 10\delta$</td>
</tr>
<tr>
<td>$[4\delta, 6\delta]$</td>
<td>$A_0A_1A_0A_2A_1A_3A_2A_4A_3A_4$</td>
<td>2</td>
<td>$5p/2 + 7\delta$</td>
</tr>
<tr>
<td>$[6\delta, 8\delta]$</td>
<td>$A_0A_1A_2A_3A_4$</td>
<td>1</td>
<td>$2p + 10\delta$</td>
</tr>
<tr>
<td>$[8\delta, 10\delta]$</td>
<td>$A_0A_3A_2A_1A_4A_3A_2A_4A_3A_0A_4A_1$</td>
<td>2</td>
<td>$5p/4 + 4\delta$</td>
</tr>
<tr>
<td>$[10\delta, 12\delta]$</td>
<td>$A_0A_4A_3A_1A_0A_4A_2A_1A_0A_3A_2A_1A_4A_3A_2$</td>
<td>3</td>
<td>$4p/3 + 14\delta/3$</td>
</tr>
<tr>
<td>$\geq 12\delta$</td>
<td>$A_0A_4A_3A_2A_1$</td>
<td>1</td>
<td>$p + 4\delta$</td>
</tr>
</tbody>
</table>

Table 7: Optimal cycles for regular balanced 4-machine cells

Moreover, for $m \geq 4$, the cycle function verifies $K(m) \geq m - 1$. This was proved in [29] using the $(m - 1)$-cycle constructed below:

$$A_0 \ A_m \ A_{m-1} \ \ldots \ A_3 \ A_1 \ A_0 \ A_m \ A_{m-1} \ \ldots \ A_4 \ A_2 \ A_1 \ A_0 \ A_m \ A_{m-1} \ \ldots \ A_3 \ A_2 \ A_1 \ A_0 \ A_m \ A_{m-1} \ \ldots \ A_4 \ A_3 \ A_2 \ A_1 \ A_m \ A_{m-1} \ \ldots \ A_4 \ A_3 \ A_2$$
For instance, for \( m = 5 \), this leads to the following 4-cycle:

\[
A_0A_5A_4A_3A_1A_0A_5A_4A_2A_1A_0A_5A_3A_2A_1A_0A_4A_3A_2A_1A_5A_4A_3A_2A_1
\]

Indeed, while executing this cycle, the robot transports \( m - 1 \) parts through the cell, but does not have the time (because of the no-wait constraint) to transport an additional part (which would have given the 1-cycle \( \pi_d \)). Using this idea, in [32], a conjecture is given on a best (optimal and with minimum degree) \( k \)-cycle in a balanced cell. This conjecture, which goes a step further for solving Agnetis’ Conjecture, is still open but some special cases have been solved. Table 8 summarizes the open cases (grey cells) in a regular balanced 5-machine cell.

<table>
<thead>
<tr>
<th>( p )</th>
<th>minimal degree of an optimal cycle</th>
<th>optimal cycle length</th>
</tr>
</thead>
<tbody>
<tr>
<td>([0, 4\delta])</td>
<td>1</td>
<td>(5p + 12\delta)</td>
</tr>
<tr>
<td>([4\delta, 8\delta])</td>
<td>2</td>
<td>(6p/2 + 8\delta)</td>
</tr>
<tr>
<td>([8\delta, 10\delta])</td>
<td>3</td>
<td>(7p/3 + 20\delta/3)</td>
</tr>
<tr>
<td>([10\delta, 12\delta])</td>
<td>3</td>
<td>(6p/3 + 18\delta/3)</td>
</tr>
<tr>
<td>([12\delta, 14\delta])</td>
<td>4</td>
<td>(6p/4 + 20\delta/4)</td>
</tr>
<tr>
<td>([14\delta, 16\delta])</td>
<td>4</td>
<td>(5p/4 + 18\delta/4)</td>
</tr>
<tr>
<td>(\geq 16\delta)</td>
<td>1</td>
<td>(p + 4\delta)</td>
</tr>
</tbody>
</table>

Table 8: Optimal cycles for regular balanced 5-machine cells

In [32], it is also proved that, for each value of \( k \) between 1 and \( \frac{m+1}{4} \), there exists an optimal \( k \)-cycle that strictly dominates all cycles with smaller degree. This implies that \( K(m) \) can not be bounded by a constant.

In the additive no-wait case, finding the best 1-cycle or 2-cycle can be done in polynomial time [27, 15]. However, the preceding discussion indicates that 1- or 2-cycles are not optimal for \( m \geq 4 \). Therefore, the complexity of finding the optimal production cycle is still an open problem. If Agnetis’ Conjecture is true, this problem is reduced to finding the best \( k \)-cycle with \( k \leq m - 1 \).

Some extensions of this problem have been studied: In [30], the combined case with some machines with the no-wait constraints and others with unbounded waiting times is considered. For balanced 2- and 3-machine cells the complete dominance table is known. For this case, \( K(2) = 1 \) and \( K(3) = 2 \) as for the no-wait case.

5 Open questions

This document has presented a state-of-the-art of the scheduling problem of identical parts in a robotic cell. In this section, we list the main questions that are still open. For unbounded waiting times, the 3-machine case is completely settled. Therefore, one can concentrate on the remaining questions on the 4-machine case (hoping that it will be possible to generalize the proofs to \( m \) machines):

- Prove that \( K(4) = 4 \) in the additive regular or in the constant case: It is only known
that, for certain instances, some 4-cycles strictly dominate cycles with smaller degree but the dominance of 4-cycles is not proved.

• Determine the exact performance factor of 1-cycles in a 4-machine cell.

More general questions also remain for additive or constant cells:

• Find the link between additive and constant cells (same results) or a metric generalizing them with the same properties (complexity of finding the best 1-cycle, identical $K(m)$ for all sub-configurations...).

• Determine lower and finite upper bounds for $K$.

• Determine the complexity of finding the best $k$-cycle with $k \leq K(m)$ in an $m$-machine cell.

For special configurations, the two main questions are:

• Determine $K(3)$ for circular cells: little work has been done on this configuration and even the 3-machine case is still open. This is linked to the preceding question that tries to generalize the metrics.

• Prove that $K(m) = 1$ for additive regular balanced cells and $m \geq 16$; even for this very simple (4 numbers) problem, the dominance of 1-cycles is not proved.

The no-wait case seems to be easier to settle since the problem is much more constrained than the case with unbounded waiting times. However, the problem of finding an optimal cycle is still open. The simpler configuration is the balanced problem (with equal processing times on all machines). Therefore, we suggest to start with this case for the following questions and then to extend them to the general case.

• Prove (or refute) Agnetis’ conjecture. This reduces to proving that $K(m) \leq m - 1$.

• Determine the complexity of finding the best $k$-cycle with $k \leq m - 1$ since we know that one has to consider cycles with degree at least until $m - 1$.

References


Les Cahiers Leibniz

Le Laboratoire Leibniz est fortement pluridisciplinaire. Son activité scientifique couvre un large domaine qui comprend des thèmes fondamentaux aussi bien en informatique qu'en mathématiques, avec une ouverture sur l'apprentissage machine, la modélisation de systèmes complexes adaptatifs, et les applications aux environnements informatiques pour l'apprentissage humain.

Les Cahiers Leibniz ont pour vocation la diffusion de rapports de recherche, de supports de cours, de textes de séminaires ou de projets de publications réalisés par des membres du laboratoire. Ils peuvent accueillir aussi des textes de chercheurs n'appartenant pas au laboratoire Leibniz mais qui travaillent sur des thèmes proches et ne disposent pas de tels supports de publication. Ces chercheurs sont priés de contacter un des membres du comité éditorial ; le comité décidera de l'acceptation du texte proposé.

Le contenu des textes publiés dans les Cahiers Leibniz relève de la seule responsabilité de leurs auteurs.

The research at Laboratoire Leibniz is multidisciplinary. It covers a large domain of fundamental and applied subjects in informatics an mathematics, with openings to machine learning, the modelisation of adaptive complex systems, and applications to teaching software.

The Cahiers Leibniz aim at diffusing research reports, lectures, and texts of conferences or pre-prints of the members of the laboratory. Moreover, the Cahiers welcome manuscripts of researchers belonging to other laboratories, working on subjects close to ours, but not disposing of such a medium. These researchers should contact one of the members of the editorial board; the latter will decide whether to accept the proposal.

The responsibility of the contents of the Cahiers lies exclusively with the authors.

Comité éditorial
Mirta B. Gordon (responsable), Annie Bessot, Gerd Finke, Humbert Fiorino, Denise Grenier, Philippe Jorrand, Andras Sëbo

Directeur de la publication
Nicolas Balacheff

Réalisation : Jacky Coutin
ISSN : 1298-020X - © laboratoire Leibniz