ELSEVIER

Operations Research Letters 27 (2000) 185–192

operations research letters

www.elsevier.com/locate/dsw

Part sequencing in three-machine no-wait robotic cells

A. Agnetis[a, *], D. Pacciarelli[b]

[a] *Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Buonarroti 12, 00185 Roma, Italy*
[b] *Dipartimento di Informatica e Automazione, Università di Roma Tre, via della Vasca Navale 79, 00146 Roma, Italy*

Received 1 September 1998; received in revised form 1 April 2000

Abstract

A no-wait robotic cell is an automated flow shop in which a robot is used to move the parts from a machine to the next. Parts are not allowed to wait. We analyze the complexity of the part sequencing problem in a robotic cell with three machines, for different periodical patterns of robot moves, when the objective is productivity maximization.

Keywords: Robotic cells; No-wait flow shop; Polynomial algorithms; NP-complete problems

1. Introduction

The *flow shop* scheduling problem consists of sequencing parts on m machines. Each machine can process only one part at a time, and all parts must visit the m machines in the same order, i.e., $M_1, M_2, \ldots, M_m$. A *robotic cell* (Fig. 1) is a flow shop which also includes an input station (M_0), an output station (M_{m+1}), and one robot, which is in charge of moving the parts from each machine to the next, and between the machines and the input/output stations. Most of the existing literature on this subject has been devoted to cells without buffers in which parts are allowed to wait on a machine, even after the completion of the processing on that machine, if either the next machine or the robot are not available. These are referred to as *bufferless robotic cells*.

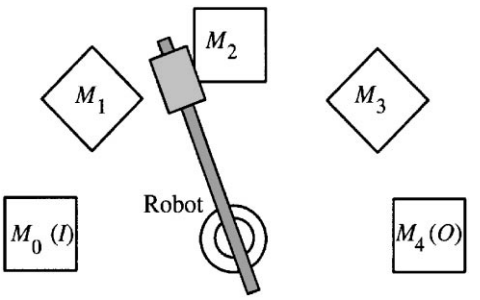

Fig. 1. The layout of a robotic cell.

In this paper we focus on the problem arising when the parts must satisfy the *no-wait constraint*, i.e., as soon as a part is completed by a machine, the robot must immediately move it to the next machine (or to the output station). In other words, machines are

* Corresponding author. Fax: +39-6-48299218.
E-mail addresses: agnetis@dis.uniromal.it (A. Agnetis), pacciare@dia.uniroma3.it (D. Pacciarelli).

0167-6377/00/$ - see front matter
PII: S0167-6377(00)00046-8

not allowed to act as buffers. We refer to this case as *no-wait robotic cells*.

Management problems in robotic cells consist in concurrently finding a schedule for *parts* and *robot moves* which maximizes productivity. For two-machine cells, the problem can be efficiently solved (see [5] for bufferless cells, and [1] for no-wait cells). For cells with three or more machines, when the parts are different from each other, the overall management problem is very hard. In fact, for both bufferless and no-wait robotic cells, if the time required by robot movement is negligible, the problem reduces to a pure part sequencing problem, which is by itself NP-hard [8]. On the other hand, it may not be practical for the robot to perform complex sequences of moves, but rather it is convenient to cyclically repeat short sequences of moves. For these reasons, some authors have especially analyzed *one-unit cycles*, i.e., patterns of robot moves during which the robot unloads (or, equivalently, loads) each machine exactly once.

When parts are all *identical*, the optimal one-unit cycle in a no-wait m-machine robotic cell can be found in polynomial time [6,7]. In [1] special results are derived for $m = 3$, showing that the optimal cycle is either a one-unit or a two-unit cycle.

In this paper we address the problem of finding the optimal one-unit cycle in a three-machine no-wait robotic cell when the parts are *different* from each other. In particular, for each of the six possible one-unit cycles, we investigate the complexity of the resulting part sequencing problem. For bufferless robotic cells, Hall et al. [5] carried out a similar study, showing that for two out of six one-unit cycles the part sequencing problem is NP-hard, and providing polynomial algorithms for the remaining four cycles. From the part sequencing viewpoint, a major difference between bufferless and no-wait cells is the following. In the bufferless case, for any fixed one-unit cycle, all part sequences are feasible. In the no-wait case, for some fixed one-unit cycles there may not even be feasible solutions, due to the fact that the robot must always reach a part on a machine no later than the end of processing on that machine. Hence, for each one-unit cycle we will be concerned with a *feasibility* problem and an *optimization* problem. We show that in four out of six cases the feasibility problem is polynomial, and in the remaining two is NP-complete. For what concerns finding the optimal part sequencing, in two cases the problem is polynomial, in another two it is NP-complete, and in the remaining two the problem is still open. In Section 2, definitions and notation are introduced. In Section 3, the six one-unit cycles are analyzed.

2. Definitions and notation

We consider robotic cells consisting of three machines, M_1, M_2, M_3, while M_0 and M_4 denote the input and output stations, respectively. The robot performs *part transfers* and *empty movements* throughout the cell. To transfer a part from machine M_i to machine M_{i+1}, the robot must consecutively unload a part from M_i, travel from M_i to M_{i+1}, and load the part on M_{i+1}. Such a part transfer operation is called *activity* A_i [2], its length being denoted by $c_{i,i+1}$, $i = 0, 1, 2, 3$. We denote by d_{ij} the time the *empty* robot takes to move from machine M_i to M_j, for $i, j = 0, \ldots, 4$ (with $i \neq j$).

The robot movements are completely specified by the order in which activities are carried out. A *cycle* is a feasible sequence of activities that the robot indefinitely repeats (on different parts). A *one-unit cycle* is a cycle in which each activity $A_0, \ldots, A_3$ appears exactly once (and hence exactly one part is produced). Since we can always describe a one-unit cycle as starting with A_0, there are six possible one-unit cycles, namely [9]

S_1: $\{A_0, A_1, A_2, A_3\}$, S_2: $\{A_0, A_2, A_1, A_3\}$,

S_3: $\{A_0, A_1, A_3, A_2\}$,

S_4: $\{A_0, A_3, A_1, A_2\}$, S_5: $\{A_0, A_2, A_3, A_1\}$,

S_6: $\{A_0, A_3, A_2, A_1\}$.

There are n parts to be cyclically produced, i.e., we consider closed-loop schedules in which the last scheduled part is followed by the first. For a fixed one-unit cycle, a feasible sequencing of parts determines the time between the loading of the first part in two consecutive repetitions of the schedule. This is the time required to cyclically produce the n parts, and will be denoted by *span*. Part h requires three consecutive operations, denoted by O_{h1}, O_{h2}, O_{h3}, to be performed by machines M_1, M_2, M_3, respectively. The *length* of operation O_{hj} is denoted by p_{hj}. Note that, due to the

no-wait constraint, if part h enters the cell at time t, then O_{h1} starts at time $t+c_{01}$, O_{h2} starts at time $t+c_{01}+p_{h1}+c_{12}$, O_{h3} at time $t+c_{01}+p_{h1}+c_{12}+p_{h2}+c_{23}$, and at time $t+c_{01}+p_{h1}+c_{12}+p_{h2}+c_{23}+p_{h3}+c_{34}$ part h leaves the cell. The robot must therefore be available at machine M_j at time $t+\sum_{i=1}^{j}(c_{i-1,i}+p_{hi})$, $j=0,1,2,3$, in order to perform the required activities on time.

For a fixed one-unit cycle we are concerned with two problems: (i) determine whether or not a feasible sequencing of parts exists, and (ii) if there are feasible sequences, find the one having minimum span. More formally, the problems addressed in the next section are the following.

Problem $F(S_i)$. *Given a three-machine no-wait robotic cell, characterized by part transfer times* $c_{i,i+1}$, $i=0,\ldots,3$, *and empty robot movement times* d_{ij}, $i,j=0,\ldots,4$, *a set of n parts having processing times* p_{hj}, $h=1,\ldots,n$, $j=1,2,3$, *determine a feasible sequence of the parts when the robot performs the one-unit cycle* S_i, *or prove that it does not exist.*

Problem $O(S_i)$. *Given a three-machine no-wait robotic cell, characterized by part transfer times* $c_{i,i+1}$, $i=0,\ldots,3$ *and empty robot movement times* d_{ij}, $i,j=0,\ldots,4$, *a MPS of n parts having processing times* p_{hj}, $h=1,\ldots,n$, $j=1,2,3$, *find, if it exists, a feasible sequence of the parts so that the span is minimum when the robot performs the one-unit cycle* S_i.

In what follows, let σ denote a sequence of parts and $\sigma(k)$ the kth part in the sequence σ. (Note that $\sigma(n+1)=\sigma(1)$.)

3. Complexity of part sequencing for the six one-unit cycles

3.1. Cycle S_1

This case (Fig. 2) is trivial, since all sequences are feasible and have the same span. The span $T(S_1)$ is simply given by the sum of the times spent in the cell by all parts, plus the time taken by the robot to move from M_4 back to M_0 after the completion of each part:

$$T(S_1)=\sum_{h=1}^{n}\sum_{j=1}^{3}p_{hj}+n(c_{01}+c_{12}+c_{23}+c_{34}+d_{40}).$$

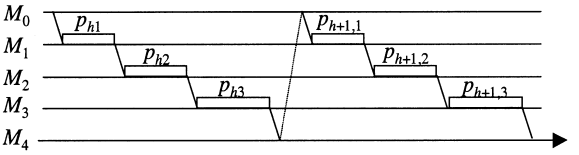

Fig. 2. A feasible schedule for cycle S_1: $\{A_0,A_1,A_2,A_3\}$.

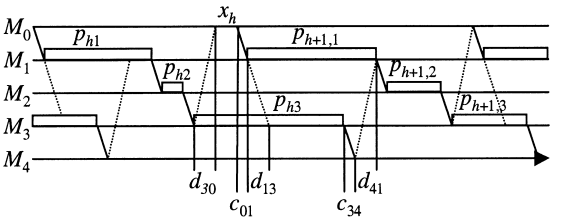

Fig. 3. A feasible schedule for cycle S_4: $\{A_0,A_3,A_1,A_2\}$.

3.2. Cycle S_4

In this section we show that both problems $F(S_4)$ and $O(S_4)$ are easy. Let us first consider $F(S_4)$. During the execution of each one-unit cycle, the robot performs A_3 on $\sigma(k)$ between activities A_0 and A_1 on $\sigma(k+1)$ (see Fig. 3). Hence, for any two consecutive parts $(\sigma(k),\sigma(k+1))$, the only overlapping operations are $O_{\sigma(k),3}$ and $O_{\sigma(k+1),1}$. In particular, after performing A_2 on part $\sigma(k)$, the robot must move to M_0, perform A_0 (on $\sigma(k+1)$) and move back to M_3 on time for the end of the operation $O_{\sigma(k),3}$. This implies that each part h must satisfy the following condition:

$$p_{h3}\geqslant d_{30}+c_{01}+d_{13}. \tag{1}$$

Similarly, we observe that after performing A_0 on part $\sigma(k+1)$, the robot must move to M_3, perform A_3 (on $\sigma(k)$) and move back to M_1 on time for the end of the operation $O_{\sigma(k+1),1}$, and this leads to the following condition:

$$p_{h1}\geqslant d_{13}+c_{34}+d_{41}. \tag{2}$$

As a consequence of the following lemma, problem $F(S_4)$ can be solved in linear time.

Lemma 1. *Given an instance of* $F(S_4)$, *a feasible sequence exists if and only if conditions* (1) *and* (2) *are satisfied for all* $h=1,\ldots,n$.

Proof. The "only if" part is implied by the above discussion. For what concerns the "if" part, it is sufficient

to observe that, given *any* sequence, it is straightforward to get a feasible schedule. If t_k denotes the completion time of operation $O_{\sigma(k),3}$, just let part $\sigma(k+1)$ enter the cell at time $t_k - d_{13} - c_{01}$. This is certainly feasible, due to condition (1). □

Let us now turn to $O(S_4)$. From now on, we suppose that the conditions in Lemma 1 are satisfied. Given a feasible schedule, consider two consecutive parts, $\sigma(k)$ and $\sigma(k+1)$. After performing A_2 on $\sigma(k)$, the robot moves to M_0. Let x_k denote the idle time spent by the robot in M_0 before starting activity A_0 on $\sigma(k+1)$. We can then write the span of the schedule as

$$\sum_{k=1}^{n} x_k + \sum_{k=1}^{n} (c_{01} + p_{\sigma(k),1} + c_{12} + p_{\sigma(k),2} + c_{23} + d_{31}). \tag{3}$$

Hence, in order to solve $O(S_4)$, we must minimize the fist term in (3), since the other does not depend on the schedule. Given σ, we observe that $O_{\sigma(k+1),1}$ cannot start too early, since the robot must have enough time to go to M_3, possibly wait for the end of $O_{\sigma(k),3}$, perform A_3 and be back in M_1 on time for the end of $O_{\sigma(k+1),1}$. In other words, it must hold (see Fig. 3):

$$d_{30} + x_k + c_{01} + p_{\sigma(k+1),1} \geqslant p_{\sigma(k),3} + c_{34} + d_{41}.$$

Hence, the minimum value x_k can attain is given by

$$x_k = \max\{0,\ p_{\sigma(k),3} + c_{34} + d_{41} - (d_{30} + c_{01} + p_{\sigma(k+1),1})\}. \tag{4}$$

In conclusion, expression (4) shows that in order to minimize the span we must solve an instance of a TSP problem in which the cost for going from city r to city s is given by $\max\{0, p'_{r2} - p'_{s1}\}$, where $p'_{r2} = p_{r3} + c_{34} + d_{41}$ and $p'_{s1} = p_{s1} + c_{01} + d_{30}$, for all $r, s = 1, \ldots, n$. This cost structure is the same occurring in an instance of two-machine no-wait flow shop with n jobs, in which job h has processing times p'_{h1} and p'_{h2} on the first and second machine respectively. Problem $O(S_4)$ can therefore be solved in $O(n \log n)$ by means of the well-known algorithm by Gilmore and Gomory [4].

3.3. Cycle S_2

In this section we establish the complexity of $F(S_2)$ (see Fig. 4). This problem turns out to be

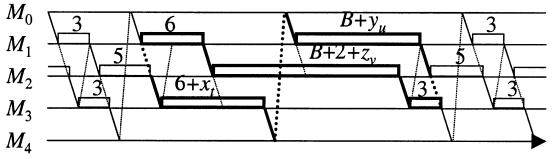

Fig. 4. A feasible schedule for cycle S_2: $\{A_0, A_2, A_1, A_3\}$.

NP-complete, and this implies the NP-hardness of $O(S_2)$. We use the following NP-complete problem [3]:

Numerical matching with target sums (NMTS). *Let* $X = \{x_1, \ldots, x_s\}$, $Y = \{y_1, \ldots, y_s\}$, $Z = \{z_1, \ldots, z_2\}$ *be sets of positive integers, such that* $\sum_{h=1}^{s} z_h = \sum_{h=1}^{s} x_h + \sum_{h=1}^{s} y_h$. *Does there exist a partition of* $X \cup Y$ *into* s *disjoint pairs* $P_1 = (x^1, y^1), \ldots, P_s = (x^s, y^s)$, *each containing exactly one element from* X and Y *and such that* $x^i + y^i = z_i$ *for* $i = 1, \ldots, s$?

Theorem 1. *Problem* $F(S_2)$ *is unary NP-complete.*

Proof. Clearly, $F(S_2)$ is in NP. We next reduce NMTS to $F(S_2)$. Given an instance of NMTS, we define the following instance of $F(S_2)$. Let $c_{i,i+1} = d_{ij} = 1$ for all i, j. There are $3s$ parts, divided into three groups of s parts, namely X-parts, Y-parts and Z-parts. The processing times of the parts are defined as follows, for the three groups respectively, for $h = 1, \ldots, s$:

$$p^x_{h1} = 3, \quad p^x_{h2} = 5, \quad p^x_{h3} = 6 + x_h,$$
$$p^y_{h1} = B + y_h, \quad p^y_{h2} = 5, \quad p^y_{h3} = 3,$$
$$p^z_{h1} = 6, \quad p^z_{h2} = B + 2 + z_h, \quad p^z_{h3} = 3,$$

where $B = 7 + \max_h\{x_h\}$. The thesis will follow from a set of intermediate claims on the structure of a feasible schedule.

Claim 1. *If an operation has length* 3 *on* M_1 *or* M_3, *or if an operation has length* 5 *on* M_2, *then in a feasible schedule the robot can never be idle during its execution. This claim follows from the observation that during the execution of an operation by* M_2, *the robot must perform three empty moves and two activities* (A_3 *and* A_0), *whereas during the execution of an operation by* M_1 *or* M_3, *the robot must perform two empty moves and one activity* (A_2 *or* A_1, *respectively*).

Claim 2. *If a feasible schedule exists, each Y-part is followed by an X-part. Consider the second and third operations of a Y-part, call them O^y_{h2} and O^y_{h3}. From the previous claim, it follows that from the start of O^y_{h2} to the end of O^y_{h3} the robot can never stay idle. So, if the next part is not an X-part, its first operation is longer than* 3, *and hence the robot would wait at M_1 for the end of such operation during the execution of O^y_{h3} (see Fig.* 4). *On the other hand, one can immediately verify that scheduling an X-part after a Y-part does not violate feasibility.*

Claim 3. *If a feasible schedule exists, each X-part is followed by a Z-part. Let O^x_{h3} indicate the third operation of the X-part. Let us first show that the next cannot be another X-part. If the next part is an X-part, the robot would be back in M_3 strictly earlier than the completion of O^x_{h3}, thus being idle for some time during the execution of the second operation of the next part, but this is unacceptable because of Claim* 1. *On the other hand, the next part cannot be a Y-part. In fact, the first operation of a Y-part is strictly longer than O^x_{h3}, and therefore, after performing A_1 on the next part, the robot would be in M_3 after the end of O^x_{h3}. Finally, since O^x_{h3} is longer than the first operation of a Z-part, scheduling a Z-part after an X-part does not violate feasibility.*

As a consequence of the above claims, and since there are exactly s parts in each group, any feasible schedule has the structure …Y–X–Z–Y–X–Z… . We are now in the position of showing that a feasible schedule exists if and only if the instance of NMTS is feasible. Given a feasible schedule, consider a subsequence of parts X–Z–Y, corresponding to the integers x_t, z_v, y_u, respectively. In the time interval from the beginning of the first and the end of the third operation on the Z-part, the robot must perform activities A_2 and A_3 on the X-part, A_0 and A_1 on the Y-part, besides A_1 and A_2 on the Z-part. Due to the feasibility of the schedule, the robot must be in M_3 no later than the end of the third operation on the Z-part, and therefore (see Fig. 4)

$$6 + c_{12} + (B + 2 + z_v) + c_{23} + 3 \geqslant d_{12} + c_{23} + (6 + x_t) + c_{34} + d_{40} + c_{01} + (B + y_u) + c_{12} + d_{23} \quad (5)$$

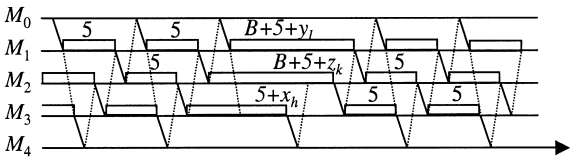

Fig. 5. A feasible schedule for cycle $S_6 : \{A_0, A_3, A_2, A_1\}$.

recalling that $c_{i,i+1} = d_{ij} = 1$ for all i, j, expression (5) yields

$$z_v \geqslant x_t + y_u, \quad (6)$$

since expression (6) must hold for all the s subsequences X–Z–Y, and since $\sum_{h=1}^{s} z_h = \sum_{h=1}^{s} x_h + \sum_{h=1}^{s} y_h$, all the (6) are indeed equalitites and a numerical matching exists. This construction also shows that if a numerical matching exists, a feasible schedule can be built having the same structure. □

3.4. Cycle S_6

In this section we establish the NP-completeness of $F(S_6)$ (see Fig. 5), and hence of $O(S_6)$.

Theorem 2. *Problem $F(S_6)$ is unary NP-complete.*

Proof. Clearly, $F(S_6)$ is in NP. We next reduce NMTS to $F(S_6)$. Given an instance of NMTS, we define the following instance of $F(S_6)$. Let $c_{i,i+1} = d_{ij} = 1$ for all i, j. Therer are $3s$ parts, divided into three groups of s parts, namely X-parts, Y-parts and Z-parts. The processing times of the parts are defined as follows, for the three groups respectively, for $h = 1, \ldots, s$:

$$p^x_{h1} = 5, \quad p^x_{h2} = 5, \quad p^x_{h3} = 5 + x_h,$$
$$p^y_{h1} = B + 5 + y_h, \quad p^y_{h2} = 5, \quad p^y_{h3} = 5,$$
$$p^z_{h1} = 5, \quad p^z_{h2} = B + 5 + z_h, \quad p^z_{h3} = 5,$$

where $B = \max_h\{x_h\}$. Again, let us suppose that a feasible schedule exists. We show that then a numerical matching exists. Let us first state some intermediate results.

Claim 1. *If an operation has length* 5, *then in a feasible schedule the robot can never be idle during its execution. This follows from the observation that during the execution of any operation, the robot must perform three empty moves and two activities.*

Claim 2. *If a feasible schedule exists, each X-part is followed by a Z-part. Let O^x_{h3} be the third operation of the X-part, of length $5 + x_h$. Note that the robot reaches M_3 to unload the X-part strictly before the end of O^x_{h3} (Fig. 5). If the next is also an X-part, the robot would be idle during the execution of the second operation of the next X-part, and this is impossible because of Claim 1. Suppose now that the next is a Y-part, and let $B + 5 + y_l$ be the length of its first operation. During the execution of O^x_{h3}, the robot performs two activities (A_1 and A_0) and three empty movements, and it must wait $B + y_l$ time units at M_1 (see Fig. 5). Since $5 + B + y_l > 5 + x_h$, the robot cannot be back in M_3 in time for the end of O^x_{h3}. It is straightforward to verify that scheduling a Z-part after an X-part does not violate feasibility.*

In the remainder of the proof, we let O^x_{h3} and O^z_{k2} denote the third and second operations of the X-part and the Z-part in a subsequence X–Z, and we indicate by $5 + x_h$ and $5 + B + z_k$ their lengths. We also denote by t_k the starting time of O^z_{k2}.

Claim 3. *If a feasible schedule exists, each subsequence X–Z is followed by a Y-part. After performing A_1 on the Z-part, the robot goes to M_0 to perform A_0 on the next part. The idle time of the robot in M_0 cannot exceed x_h, since otherwise the robot does not get to M_3 before the end of O^x_{h3}. So, the first operation of the next part cannot start later than $t_k + 2 + x_h$. On the other hand, after the end of O^z_{k2}, the robot cannot be in M_1 before $t_k + 5 + B + z_k + 2$. Hence, since the schedule is feasible, the first operation of the next part must be strictly greater than 5, i.e., the next part is necessarily a Y-part.*

As a consequence of the above claims, and since there are exactly s parts in each group, any feasible schedule has the structure …X–Z–Y–X–Z–Y… . We are now in the position of showing that a feasible schedule exists if and only if the instance of NMTS is feasible. Given a feasible schedule, consider a subsequence of parts X–Z–Y, corresponding to the integers x_h, z_k, y_l, respectively. From Claim 3, the first operation of the Y-part cannot end later than $t_k + 2 + x_h + 5 + B + y_l$, while the robot does not get to M_1 before $t_k + 5 + B + z_k + 2$. Since on the other hand the robot must get to M_1 no later than the end of the first operation of the Y-part, it follows that

$$z_k \leqslant x_h + y_l, \tag{7}$$

since expression (7) must hold for all the s subsequences X–Z–Y, and since $\sum_{h=1}^{s} z_h = \sum_{h=1}^{s} x_h + \sum_{h=1}^{s} y_h$, all the (7) are indeed equalities and a numerical matching exists. This construction also shows that if a numerical matching exists, a feasible schedule can be built having the same structure. □

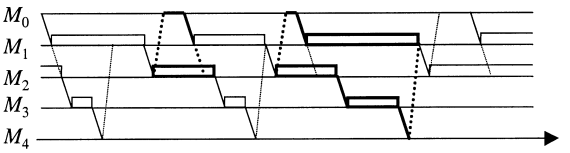

Fig. 6. A feasible schedule for cycle S_5: $\{A_0, A_2, A_3, A_1\}$.

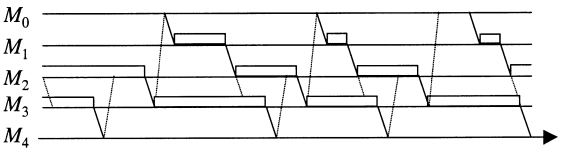

Fig. 7. A feasible schedule for cycle S_3: $\{A_0, A_1, A_3, A_2\}$.

3.5. Cycles S_3 and S_5

In this section we show that problems $F(S_3)$ and $F(S_5)$ are easy. Let us first consider $F(S_5)$.

During the execution of each one-unit cycle, the robot must perform A_0 on $\sigma(k+1)$ before activity A_2 on $\sigma(k)$ (see Fig. 6). On the other hand, A_1 on $\sigma(k+1)$ must be executed after A_3 on $\sigma(k)$. As in Section 3.2, x_k denotes the idle time spent by the robot in M_0 before A_0 on $\sigma(k+1)$. For a sequencing σ to be feasible, the following conditions must be satisfied by each pair of consecutive parts (see Fig. 6).

$$p_{\sigma(k),2} \geqslant d_{20} + x_{\sigma(k)} + c_{01} + d_{12},$$

$$p_{\sigma(k),2} + c_{23} + p_{\sigma(k),3} + c_{34} + d_{41} \leqslant d_{20} + x_{\sigma(k)} + c_{01} + p_{\sigma(k+1),1},$$

since $x_{\sigma(k)}$ is nonnegative, these are equivalent to

$$p_{\sigma(k),2} \geqslant d_{20} + c_{01} + d_{12}, \tag{8}$$

$$c_{23} + p_{\sigma(k),3} + c_{34} + d_{41} \leqslant p_{\sigma(k+1),1} - d_{12}. \tag{9}$$

Table 1
Complexity of no-wait and bufferless problems

Cycle	Feasibility no-wait	Optimality no-wait	Optimality bufferless
S_1	Trivial	Trivial	Trivial
S_2	NPC	NP-hard	NP-hard
S_3	$O(n \log n)$	Open	$O(n \log n)$
S_4	$O(n)$	$O(n \log n)$	$O(n \log n)$
S_5	$O(n \log n)$	Open	$O(n \log n)$
S_6	NPC	NP-hard	NP-hard

The necessary condition (8) does not depend on the sequence, and can therefore be easily checked. In what follows we assume that all parts satisfy (8). Hence, $F(S_5)$ consists of finding a sequencing σ such that condition (9) is satisfied for all $k = 1, \ldots, n$.

Consider an auxiliary instance of no-wait flow shop with n jobs and two machines, in which the processing times on the two machines are, respectively, $p'_{h1} = p_{h1} - d_{12}$ (which is nonnegative due to (8)) and $p'_{h2} = c_{23} + p_{h3} + c_{34} + d_{41}$. We recall that the no-wait flow shop problem with two machines is a special case of the TSP problem in which the cost for going from city r to city s is given by $\max\{0, p'_{r2} - p'_{s1}\}$.

Theorem 3. *A feasible sequence for $F(S_5)$ exists if and only if the optimal solution of the auxiliary no-wait flow shop problem has value* 0.

Proof. Let $\tilde{\sigma}$ be an optimal cycle in the auxiliary TSP instance. If its value is 0, then $p'_{\tilde{\sigma}(k),2} - p'_{\tilde{\sigma}(k+1),1} \leqslant 0$ for all $k = 1, \ldots, n$. From the definition of the processing times in the auxiliary problem, it follows that $\tilde{\sigma}$ satisfies condition (9). On the other hand, if the value of an optimal cycle is strictly positive, for any sequence σ there is always at least on k for which $p'_{\sigma(k),2} - p'_{\sigma(k+1),1} > 0$, hence violating condition (9). □

As a consequence of the above theorem, $F(S_5)$ can be solved in $O(n \log n)$ by the algorithm by Gilmore and Gomory [4].

Finally, for what concerns $F(S_3)$, it is straightforward to show that this problem is symmetrical to $F(S_5)$. In fact, if we turn upside down the Gantt chart of a feasible schedule for S_5, we get a feasible schedule for S_3. For instance, Fig. 7 shows the schedule obtained in this way from the one in Fig. 6.

4. Conclusions

We presented some new results for part sequencing in three-machine no-wait robotic cells, when the robot move cycle is fixed. There are six possible robot cycles. Table 1 summarizes the state of the art for the problems analyzed in this paper, as well as for the corresponding problems in bufferless robotic cells (see [5]). Recall that no feasibility problem arises in bufferless robotic cells, since all sequences are feasible for any robot cycle.

Problem $O(S_3)$ and $O(S_5)$ are special cases of TSP (Section 3.5) displaying different features from other TSP problems arising, e.g., in three-machine bufferless cells or in no-wait flow shop. (For instance, in $O(S_3)$ triangle inequality does not hold, since the robot cycle forbids certain parts to be processed consecutively.) Future research will address the complexity of such open problem, as well as possible extensions of the results presented here to general m-machine cells.

References

[1] A. Agnetis, Scheduling no-wait robotic cells with two and three machines, European J. Oper. Res. 123 (2) (2000) 303–314.
[2] Y. Crama, J. van de Klundert, Cyclic scheduling of identical parts in a robotic cell, Oper. Res. 45 (6) (1997) 952–965.
[3] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman and co., New York, 1979.
[4] P.C. Gilmore, R.E. Gomory, Sequencing a one-state variable machine: a solvable case of the traveling salesman problem, Oper. Res. 9 (1964) 849–859.
[5] N.G. Hall, H. Kamoun, C. Sriskandarajah, Scheduling in robotic cells: classification, two and three machines, Oper. Res. 45 (3) (1997) 421–439.

[6] V.B. Kats, E. Levner, A strongly polynomial algorithm for no-wait cyclic robotic flowshop scheduling, Oper. Res. Lett. 21 (1997) 171–179.
[7] E. Levner, V.B. Kats, V.E. Levit, An improved algorithm for a cyclic robotic scheduling problem, European J. Oper. Res. 97 (1997) 500–508.
[8] H. Röck, The three-machine no-wait flow shop is NP-complete, J. ACM 31 (2) (1984) 336–345.
[9] S.P. Sethi, C. Sriskandarajah, G. Sorger, J. Blazewicz, W. Kubiak, Sequencing of parts and robot moves in a robotic cell, Internat. J. Flexible Manufacturing Systems 4 (1992) 331–358.