Cyclic scheduling in 3-machine robotic flow shops

Yves Crama [1]
Joris van de Klundert [2]

[1]Ecole d'Administration des Affaires, Université de Liège, 4000 Liège, Belgium. Email: Y.Crama@ulg.ac.be

[2]Department of Quantitative Economics, Faculty of Economics and Business Administration, Maastricht University, 6200 MD Maastricht, The Netherlands. Email: J.vandeKlundert@ke.unimaas.nl

Abstract

We consider a robotic flow shop model in which a single robot is responsible for the transportation of parts between machines. For reasons of simplicity, when the shop is to produce a large number of identical parts, the robot usually performs repeatedly a fixed sequence of activities. This sequence of activities is called a 1-unit cycle if each execution of the sequence results in the production of exactly one part. It has been conjectured that 1-unit cycles yield optimal production rates for 3-machine robotic flow shops. We establish the validity of this conjecture.

keywords : cyclic scheduling, sequencing, automated manufacturing, flow shop.

1 Introduction

In recent years, technology advances and worldwide competition have stimulated many manufacturers to invest in highly automated manufacturing systems. However, some of the issues arising in the planning and control of such systems are not extremely well understood yet. An example of this phenomenon is provided by the consideration of the impact of material handling systems (MHS) on the effici ency of automated facilities.

Classical scheduling models often ignore the constraints imposed by material handling operations (transfers, loading, unloading), thereby implicitly assuming that the MHS does not constitute a bottleneck or a limited resource (as an example, transportat ion times are assumed to be zero between the machines of a classical flow shop model). On the other hand, in automated environments, material handling is usually performed by flexible devices, such as robots or automated guided vehicles, which share a h igh degree of integration with the workstations. As a consequence, the performance of the system is directly affected by material handling activities. Therefore, the material handling system must be explicitly taken into account in scheduling models of automated facilities. This point has been recently stressed by several researchers. In particular, one of the areas in which interesting research problems have emerged is the scheduling of *robotic flow shops*: see e.g. Asfahl [1985], Crama [1995], Sethi et al. [1992], van de Klundert [1996].

For our purpose, a robotic flow shop may be viewed as a line of machines and input/output devices in which materials handling is performed by a single robot. In order to simplify control, the robot is often restricted in practice to perform a fixed sequence of elementary activities (load machine M_i, move to machine M_j, unload it, ...) which is repeated a large number of times until all parts of a given type have been produced (see e.g. Asfahl [1985]). This sequence of activities is called a 1-*unit cycle* if each execution of the sequence results in the production of exactly one part. From a theoretical point of view, 1-unit cycles are known to yield suboptimal throughput rates in most production settings. However, Sethi et al. [1992] conjectured that, in 3-machine no-buffer robotic flow shops, 1-unit cycles yield optimal throughput rates when all the parts to be produced are identical. Hall et al. [1993] and Sriskandarajah et al. [1994] provided some evidence for this conjecture. Their approach could conceivably be generalized to produce more elements of evidence, but it is – by nature – computationally burdensome and provides little insight into the conjecture.

In this paper, we present a complete proof of the conjecture. Our approach is based on a novel, compact representation of the state space of the cell, which allows us to capture the relevant features of all possible sequences of robot moves while reducing the number of cases to be considered in the proof.

In the next section, we define more precisely the robotic flow shop scheduling problems that we want to address. In this discussion we pay special attention to issues concerning the consequences of restricting the set of allowable sequences of robot activities. In Section 3, we present the state space graph model that facilitates our analysis of the conjecture. In Section 4 we prove the conjecture. Section 5 provides a brief discussion of our results and outlines future research directions.

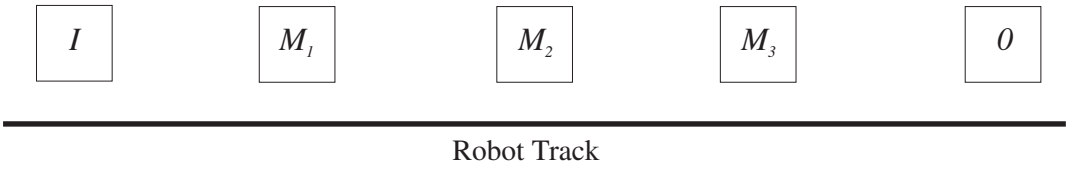

Figure 1: A 3-machine robotic cell (line layout)

2 Robotic flow shops

2.1 Overview of the model

We first describe our model of a robotic flow shop (Crama [1995], Hall et al. [1993] and van de Klundert [1996] provide a more complete description of this and other models). A robotic flow shop consists of m machines M_0 to M_m, an input device denoted I or M_0, an output device denoted O or M_{m+1}, and a robot. The parts to be produced enter the flow shop at the input device, are successively processed on machines $M_1, M_2, \ldots, M_m$, in this order, and are finally delivered at the output device (see Figure 1). The robot performs two types of tasks. On the one hand, it transports the parts from machine t o machine. On the other hand, it loads the parts onto the machines for processing and unloads them after processing. We assume that there are no buffers or storage facilities between the machines, as is often the case in modern, lean production environments. Thus, at any instant, each part is located either at the input device, or at the output device, or on a machine , or is being handled by the robot. Further, the machines M_1 to M_m as well as the robot can only handle one part at a time. Each machine starts processing as soon as it is loaded, but remains idle after it has processed a part until the robot has unloaded it and loaded it again with the next part. This situation is akin to that encountered in traditional flow shop scheduling with *blocking* (see Hall and Sriskandarajah [1996]).

Let us further specify the behavior of the robot. In order to model travel times, we adopt a simple and often used model in which the robot is assumed to move at constant speed along a linear track, see e.g. Figure 1. For $i, j \in \{0, \ldots, m+1\}, i \neq j$, let $\delta_{i,j} > 0$ denote the amount of time required by the robot to travel from machine M_i to machine M_j. Then, the assumption of constant speed means that, for $0 \leq i < j \leq m+1$, the time $\delta_{i,j}$ required by the robot to travel from machine M_i to machine M_j satisfies the following condition:

$$\delta_{i,j} = \delta_{j,i} = \sum_{k=i}^{j-1} \delta_{k,k+1}. \tag{1}$$

Let us now turn to the handling and processing requirements of the parts. We denote by ϵ_i the amount of time required by the robot to load or unload machine M_i, for $i =$

$0, \ldots, m+1$, and by p_i the processing time of each part on machi ne M_i, for $i = 1, \ldots, m$. Observe that we assume these handling and processing times to be *independent* of the part. In other words, all parts are *identical*.

Finally, we assume that there are infinitely many parts to be produced and that the objective of the problem is to minimize the long run average cycle time (see below for definitions). These assumptions are meant to model a common situation in contempor ary manufacturing, in which manufacturers often produce medium-size batches of identical products.

In the sequel, we denote by $RFIP$ this *robotic flow shop scheduling problem with identical parts*. To understand the difficulties linked to the presence of the robot in this setting, let us briefly review the complexity of some related scheduling problems for traditional (non robotic) no-buffer flow shop. If there are only two machines and n (distinct) parts, then the part input sequence that minimizes makes pan can be computed efficiently by a classical algorithm of Gilmore and Gomory [1964]. As observed by McCormick et al. [1994], it follows from a result of Papadimitriou and Kanellakis [1980] that minimizing cycle time is strongly NP-Complete when there are three or more machines in the shop. Observe, however, that if all parts have identical processing requirements, then the part input sequencing problem vanishes and flow shop scheduling problems become trivial regardless of the number of machines.

Unfortunately, the same conclusion cannot be drawn in the case of a robotic flow shop. Even if all parts are identical, there still remains to determine the optimal sequence of robot moves, that is the order in which the robot should serve the machines. The resulting $RFIP$ problem is highly non trivial and appears to be intere sting both from a theoretical and from a practical point of view.

2.2 Activities, robot move sequences and schedules

Let us examine more closely the sequence of tasks performed by the robot.

Definition 1 For $i = 0, \ldots, m$, the sequence of robot moves

1) unload M_i,

2) travel from M_i to M_{i+1},

3) load M_{i+1},

is called *(robot) activity* i and is denoted by A_i.

It should be clear that the sequence of moves performed by the robot while processing a batch of parts can be completely described by the corresponding sequence of robot activities (see Sethi et al. [1992]). Conversely, the question arises of which sequences of activities represent executable sequences of robot moves.

Definition 2 An infinite sequence of activities $A_{i_0}, A_{i_1}, \ldots$ is called a *feasible robot move sequence* if, for $i = 1, \ldots, m-1$, each of A_{i-1} and A_{i+1} occurs exactly once between any two consecutive occurences of A_i.

Together, the conditions in the aforementioned definition imply that the robot never has to unload an empty machine and never has to load a machine that is already loaded.

For obvious reasons of simplicity and practicality, researchers as well as practitioners have often focussed on cyclic sequences of robot activities, namely sequences that can be obtained by iterating a fixed, finite sequence of activities. Let us denot e by π^l the l-fold repetition of sequence π, where $l \in \mathbb{N} \cup \{\infty\}$.

Definition 3 For $k \in \mathbb{N}$, a *k-unit cycle* is a finite sequence π of robot activities in which each of $A_0, A_1, \ldots, A_m$ occurs exactly k times, and such that the robot move sequence π^∞ *generated by* π is feasible.

Notice that, in particular, exactly k parts are produced (i.e., are unloaded at the output device) during each execution of a k-unit cycle. Simplest and most thoroughly studied among k-unit cycles are the 1-unit cycles. From the previous definition, we see that a 1-unit cycle is a permutation of the activities $A_0, A_1, \ldots, A_m$. Interestingly, the converse statement is also true.

Lemma 1 Every permutation of the activities $A_0, A_1, \ldots, A_m$ is a 1-unit cycle.

Proof. See Lieberman & Turksen [1981], Sethi et al. [1992]. ∎

One-unit cycles will be our main object of study in forthcoming sections. As a matter of fact, the purpose of this paper is to establish the optimality of 1-unit cycles for the 3-machine robotic flow shop problem with identical parts. In order to procee d with a precise statement and a proof of this result, however, we first need to introduce a few more definitions.

Definition 4 A *schedule* for the *RFIP* problem is a function $S(A_i, t)$ that assigns a starting time to the t-th execution of activity A_i, for $i = 0, \ldots, m$ and $t \in \mathbb{N}$. The *long run average cycle time* of S is equal to

$$\limsup_{t \to \infty} \frac{S(A_m, t)}{t}.$$

We are not going to dwell here on the characterization of *feasible* schedules, which we assume to be intuitively clear (see e.g. van de Klundert [1996] for a formal definition). Let us simply observe that every such feasible schedule S yields, i n a natural way, a feasible robot move sequence σ: we say that σ is the sequence *implied by* S. The converse relation must be more carefully defined: indeed, several schedules may very well imply the same sequence of moves. Let us t herefore introduce a special category of schedules:

Definition 5 The *active schedule* associated with a robot move sequence σ is the unique schedule S implying σ in which the robot starts all moves and load/unload operations as early as possible and performs them as qu ickly as possible, i.e. S is the unique schedule such that, for every other schedule T implying σ, $S(A_i, t) \leq T(A_i, t)$ for all $i = 0, \ldots, m$ and $t \in \mathbb{N}$.

Notice that, in an active schedule, we can assume that the robot never executes any unnecessary move, such as travelling around the shop while waiting for a machine to finish processing. It is not difficult to see that the active schedule associated wit h a sequence σ has minimum long run average cycle time among all schedules implying σ. This observation motivates our next definition:

Definition 6 The *long run average cycle time* of a robot move sequence σ is the long run average cycle time of the associated active schedule. The long run average cycle time $c(\pi)$ of a k-unit cycle π is the long run average cycle time of the infinite sequence π^∞ generated by π.

We are now ready to give a formal statement of the conjecture proposed by Sethi et al. [1992].

Conjecture 1 In a 3-machine no-buffer robotic flow shop with identical parts, the minimum long run average cycle time is achieved by a 1-unit cycle.

The main goal of this paper is to provide a proof of Conjecture 1. For this purpose, it will be useful to consider first a slightly weaker version of the conjecture, which we state as follows:

Conjecture 2 In a 3-machine no-buffer robotic flow shop with identical parts, there is a 1-unit cycle that minimizes the long run average cycle time over the set of k-unit cycles, for all $k \in \mathbb{N}$.

To understand why Conjecture 2 is easier to handle than Conjecture 1, let us now introduce the concept of periodic schedule:

Definition 7 For $k \in \mathbb{N}$, a schedule S is called *k-periodic* if there exists a constant $C \in \mathbb{R}$ (called the *cycle time* of S) such that, for every $i \in \{0, \ldots, m\}$ and for every $t \in \mathbb{N}$, $S(A_i, t+k) - S(A_i, t) = kC$.

Obviously, the average long run cycle time of a periodic schedule is achieved and is equal to its cycle time C. On the other hand, van de Klundert [1996] proved:

Lemma 2 For every k-unit cycle π, the long run average cycle time of π exists and is achieved by the cycle time of a k-periodic schedule.

Proof. See Theorem 2.3 in van de Klundert [1996]. ∎

Conjecture 2 simply expresses that, among all long run cycle times that can be obtained by indefinitely repeating a single sequence of robot moves, the minimum is achieved by the cycle time of a 1-unit cycle. This statement is interesting in several respects. First, as already mentioned, 1-unit cyles are especially attractive from a practical point of view because of their conceptual simplicity and their ease of implementation. Moreover, Crama and van de Klundert [1994] proved that, for id entical parts, the optimal 1-unit cycle can be found in time polynomial in m, viz. the number of machines. Finally, Conjecture 2 will provide a convenient stepping-stone toward a proof of Conjecture 1.

We now proceed with a brief survey of previous work related to the conjectures.

2.3 Previous results

Sethi et al. [1992] showed that, when the shop consists of only two machines, i.e. when $m = 2$, then there is a 1-unit cycle that achieves minimum cycle time among all possible robot move sequences. Their reasoning is based on a state-space representat ion of the problem. For 3-machine robotic shops, Sethi et al. [1992] established that, among 1-unit cycles, one of the following four *pyramidal cycles* always is optimal (the term 'pyramidal' is introduced and motivated by Crama and van de Klunder t [1994]):

1. (A_0, A_1, A_2, A_3), to be called the *uphill cycle*,
2. (A_0, A_2, A_3, A_1),
3. (A_0, A_1, A_3, A_2),
4. (A_0, A_3, A_2, A_1) to be called the *downhill cycle*.

(This observation is generalized to arbitrary values of m by Crama and van de Klundert [1994].) These four pyramidal 1-unit cycles will also turn out to play an important role throughout the analysis required to prove the conjectures; see Section 4.

Hall et al. [1993] and Sriskandarajah et al. [1994] established some special cases of Conjecture 2. Namely, they showed that, in a 3-machine flow shop, the minimum cycle

time achievable by 1-unit cycles does not exceed the minimum cycle time achieved by 2-unit cycles and certain types of 3-unit cycles. Their proof relies on the enumeration of these sh ort cycles and on the numerical solution of an integer programming formulation of the cycle time minimization problem.

Finke et al. [1996] proved that 1-unit cycles are optimal when there is a unit-capacity output buffer available at each machine. On the other hand, several generalizations of Conjecture 2 are known to be invalid. For instance, the conjectur e does not hold for non-identical parts (Hall et al. [1993]), nor when finite upper bounds are imposed on the time that each part can spend on each machine (Lei [1995], Kats [1995]). Also, the conjectures only bear on flow shops in which the travel time of the robot is modelled by Equation 1. Indeed, Hertz [1995] provided an instance of a 3-machine robotic flow shop in which the travel times satisfy the triangle inequalit ies ($\delta_{i,j} \leq \delta_{i,k} + \delta_{k,j}$ for all i, j, k) without satisfying Equation 1, and for which 1-unit cycles are not optimal.

3 A state space graph model

For notational convenience, we restrict our analysis to the special case of $RFIP$ where the travel time between consecutive machines is constant: $\delta_{i,i+1} = \delta > 0$ for $i = 0, \ldots, m$ and where all load/unload operations are instantaneous: $\epsilon_i = 0$ for $i = 0, \ldots, m+1$. All our results can be generalized in a straightforward manner to the general $RFIP$ model described in Section 2, at the expense of heavier notations (see Section 5).

We also assume from now on that the robot only executes active schedules: clearly, as far as proving the conjectures is concerned, this assumption can be made without loss of generality.

Our proofs are based on a state space graph model of the robotic flow shop which we now proceed to explain. Observe first that, in order to completely specify the state of a robotic flow shop at any given instant, it is sufficient to communicate the fol lowing pieces of information:

1) for each machine, whether it is loaded or not;

2) for the robot, whether it is carrying a part or not;

3) for each machine, the remaining processing time of the part loaded on the machine, if there is one;

4) the position of the robot.

Formally, a *shop state* for a 3-machine flow shop is a vector $(v_1, v_2, v_3, w, r_1, r_2, r_3, pos)$ where v_i is 1 if machine M_i is loaded and v_i is 0 otherwise, w is 1 if the robot is carrying a part and w is 0 otherwise, r_i denotes the remaining processing time on machine M_i and pos indicates the position of the robot ($i = 1, 2, 3$). We denote by $\mathcal{F}$ the set of all feasible shop states. (Since we will not use them explicitly, we do not enumerate the conditions that must be fulfilled by a feasible shop state: for instance, (v_1, v_2, v_3, w) must

be in $\{0,1\}^4$, $r_i = 0$ when $v_i = 0$, etc.) The evolution of the robotic flow shop can now be viewed as tracing a 'trajectory' of shop states in $\mathcal{F}$ (see Sethi et al. [1992]).

Rather than dealing directly with the (uncountable) set $\mathcal{F}$, we will find it more convenient to handle a finite, condensed version of this set. If $(v_1, v_2, v_3, w, r_1, r_2, r_3, pos)$ is a feasible shop state, then the triple $v = (v_1, v_2, v_3)$, w hich only specifies whether each machine is loaded or not, will be called the *L/U state* of the flow shop. Notice that every vector $v \in \{0,1\}^3$ is a feasible L/U state. In order to know the evolution of the flow shop through time, it is only necessary to know the sequence of L/U states that are succesively encountered: this is actually equivalent to knowing the full sequence of robot moves, from which the associated ac tive schedule can be deduced (see Section 2.2).

In order to facilitate the presentation of the proof, we define two special sets of shop states associated with each L/U state $v \in \{0,1\}^3$. One set, denoted $e(v)$ ('entering state v'), contains all shop states that can possibly arise when the r obot has just loaded a machine and the L/U state has thereby switched to v. Formally:

$$e(v) = \bigcup_{\substack{i=1,2,3,4 \\ v_{i-1}=0, v_i=1}} \{(v, w, r, pos) \in \mathcal{F} \mid w = 0, r_i = p_i, pos = M_i\}$$

(where the conditions $v_0 = 0$, $v_4 = 1$ and $r_4 = p_4$ are considered to be identically fulfilled, by convention).

Let us briefly comment on this definition. Consider a shop state $(v, w, r, pos) \in e(v)$. If $r_i = p_i$ for some index $i \leq 3$, then it must be the case that the robot has just loaded a part on machine M_i (since the remaining processing time on M_i is p_i). Notice that this can only occur if $v_{i-1} = 0$, $v_i = 1$, $w = 0$ and $pos = M_i$ (meaning that the robot is located at M_i). On the other hand, if $w = 0$ and $pos = M_4$, then the robot is located at the output device and is not car rying any part. Since we only consider active schedules, we can conclude that the robot has just unloaded a part at the output device, so that v_3 must be equal to 0.

The second set of shop states associated with v is denoted $l(v)$ ('leaving state v'). It contains all shop states for which the L/U state is currently v, but could instantaneously switch to another state due to the unloading of some machine (poss ibly the input device):

$$l(v) = \bigcup_{\substack{i=0,1,2,3 \\ v_i=1, v_{i+1}=0}} \{(v, w, r, pos) \in \mathcal{F} \mid w = 0, r_i = 0, pos = M_i\}$$

(where the conditions $v_0 = 1$, $v_4 = 0$ and $r_0 = 0$ can be disregarded).

To interpret this definition, consider a shop state $(v, w, r, pos) \in l(v)$ and assume that, for some index $i \geq 0$, there holds $v_i = 1$, $v_{i+1} = 0$, $w = 0$, $r_i = 0$ and $pos = M_i$. Then, the robot can start executing activity A_i right away; namely, machine M_i has completely processed a part, machine M_{i+1} is available and the robot is ready to unload M_i.

We are now ready to describe the state space graph G to be used in the proof of the conjecture (see Figure 2). The vertices of G are the 16 sets $e(v), l(v)$ for $v \in \{0,1\}^3$. For

each v, we draw an arc from $e(v)$ to $l(v)$. Moreover, for ea ch $v \neq w$, we draw an arc from $l(v)$ to $e(w)$ if the flow shop can leave the L/U state v and enter the L/U state w without going through any other intermediate L/U state. Each such arc can be viewed as representing the (unique) activity which transforms the L/U state v into the L/U state w.

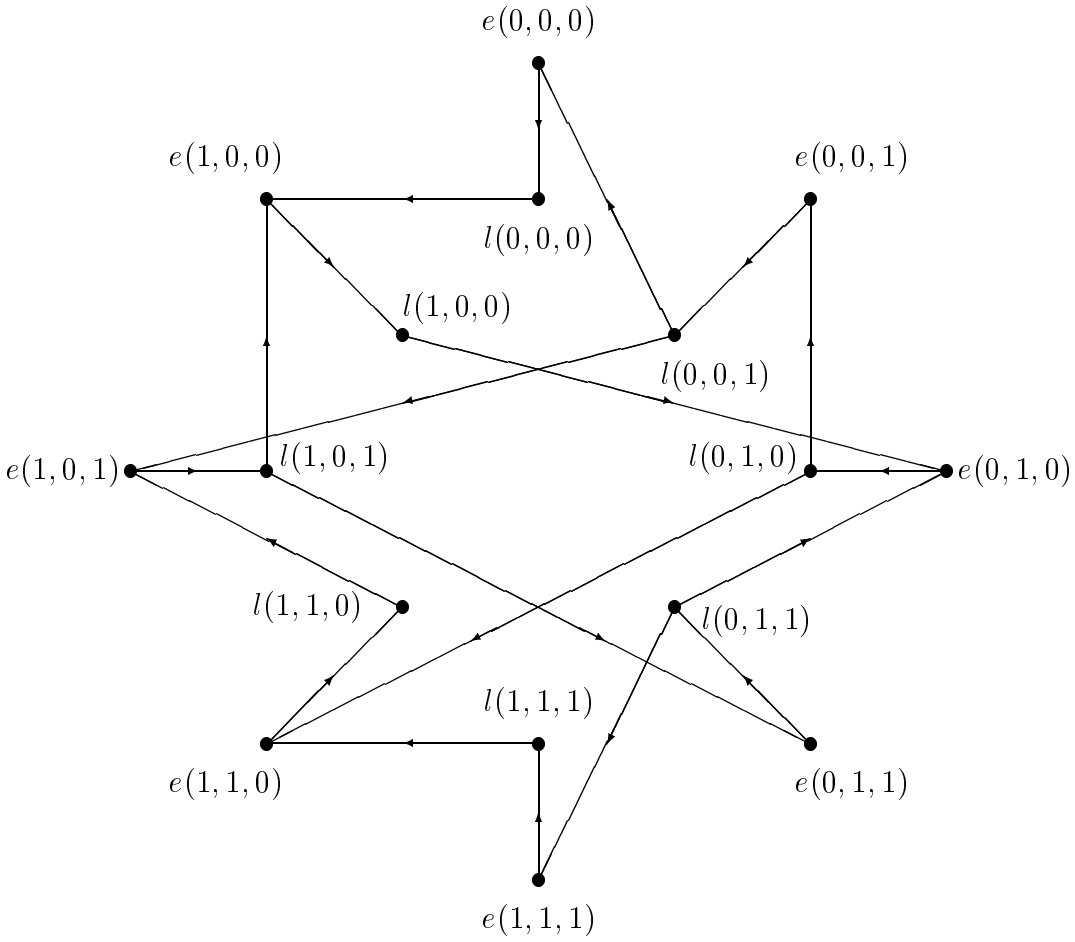

Figure 2: The state space graph

From the preceding discussion, it should be clear that there is a one-to-one correspondence between robot move sequences and directed walks in G. In addition, there is a one-to-one correspondence between k-unit cycles and directed closed walks in G consisting of $8k$ arcs. Therefore, in the sequel, we do not distinguish between robot move sequences and directed walks on the one hand, or between k-unit cycles and closed walks of length $8k$ on the other hand.

Let us introduce some shorthand notations. We will often omit parentheses and

commas when refering to L/U states: for example, we write 101 for the L/U state $(1,0,1)$. Furthermore, we use the shorthand 101 011 to denote the sequence of L/U states $((1,0,1),(0,1,1))$ or, alternatively, a sequence of successive shop states in the sets $e(1,0,1)$, $l(1,0,1)$, $e(0,1,1)$ and $l(0,1,1)$ (the precise meaning will always be clear from context). We use '/' as logical 'or'; e.g., 111/010 means L/U state 111 *or* L/U state 010. As before, π^l denotes a sequence consisting of l repetitions of the sequence π (where possibly $l = 0$ or $l = \infty$).

For instance, with these notations, the four pyramidal 1-unit cycles introduced in Section 2.3 can be represented as follows:

1. 000 100 010 001 (the uphill cycle),
2. 010 110 101 100,
3. 001 101 011 010,
4. 011 111 110 101 (the downhill cycle).

The corresponding directed cycles are easily identified in Figure 2.

In the next section, we are going to examine more closely the structure of G and of its directed walks and to derive properties which will enable us to prove Conjectures 1 and 2.

4 Proofs of the conjectures

4.1 Preliminary results

We start with a few important observations which will greatly simplify the analysis of closed walks in G and which will thereby enable us to prove the conjectures.

Consider first the set $e(0,0,1)$. For any shop state in this set, the remaining processing times are necessarily $(0,0,p_3)$ and the robot is idle at M_3. In particular,

Observation 1 $|e(0,0,1)| = 1$.

Similarly, for any shop state in $l(1,0,0)$, the robot must be positioned at M_1 and the remaining processing times are $(0,0,0)$. Therefore, we have

Observation 2 $|l(1,0,0)| = 1$.

Remark 1 There is a certain symmetry in these two observations. If the parts were to flow through the shop in the reverse direction, $e(0,0,1)$ would take the role of $l(1,0,0)$ and vice versa. We do not exploit this symmetry in the remainder of the analysis, although we think observations of this type could be helpful, e.g. in proving the optimality of 1-unit cycles for m-machine robotic flow shops.

The interpretation of Observations 1 and 2 is clear: when the shop enters the L/U state 001 or leaves the L/U state 100, the shop state is *completely* and *uniquely* determined, and we know *everything* there is to know about it. These obse rvations are helpful in that they allow to reduce the number of robot move sequences to be considered in the analysis: indeed, any sequence that leads the shop into one of the L/U states 001 or 100 can be considered as losing all memory from previous st ates and starting anew from the current state. This informal discussion will now be turned into precise statements.

Lemma 3 For all $k \geq 1$, if $e(0,0,1)$ appears in an optimal k-unit cycle, then there exists an optimal s-unit cycle (for some $s \leq k$) in which $e(0,0,1)$ appears exactly once.

Proof. Let π be a k-unit cycle (or equivalently, a directed closed walk of G) containing several occurences of $e(0,0,1)$. Decompose π into $\pi = (e(0,0,1), \sigma, e(0,0,1), \tau, e(0,0,1))$, where $\sigma^* = (e(0,0,1), \sigma, e(0,0,1))$ and $\tau^* = (e(0,0,1), \tau, e(0,0,1))$ are two subsequences of π, and σ does not contain $e(0,0,1)$. Then, σ^* and τ^* are themselves s- and t-unit cycles, for appropriate values of s and t. Moreover, Observation 1 implies that $c(\sigma^*)$ (that is, the cycle time of σ^*) depends only on σ, but not on the remainder of π. Similarly, the cycle time $c(\tau^*)$ of τ^* depends only on τ. From the optimality of π, we conclude now that $c(\pi) = c(\sigma^*) = c(\tau^*)$ (otherwise, deleting the subsequence with longest cycle time would improve π). In particular, σ^* is an opt imal cycle. ∎

Exactly the same reasoning holds for $l(1,0,0)$.

Lemma 4 For all $k \geq 1$, if $l(1,0,0)$ appears in an optimal k-unit cycle, then there exists an optimal s-unit cycle (for some $s \leq k$) in which $l(1,0,0)$ appears exactly once.

Proof. See proof of Lemma 3. ∎

Now, consider a fixed optimal k-unit cycle π. As a consequence of Lemma 3 and Lemma 4, we can distinguish between four mutually exclusive and collectively exhaustive cases:

Case 1. Neither $e(0,0,1)$ nor $l(1,0,0)$ occurs in π.

Case 2. $l(1,0,0)$ occurs (exactly once) in π, but $e(0,0,1)$ does not.

Case 3. $e(0,0,1)$ occurs (exactly once) in π, but $l(1,0,0)$ does not.

Case 4. Both $e(0,0,1)$ and $l(1,0,0)$ occur (exactly once) in π.

In Subsections 4.2-4.5, we consider successively each of these four cases. In each case, we exhibit a 1-unit subcycle of π with optimal cycle time. Interestingly, the 1-unit cycles that arise in this way are the four pyramidal cycles identified in Section 2.3.

This case analysis will be further simplified by the next dominance result, which shows that the long run cycle time *cannot* increase when we replace a subsequence (101 011 010 110) by the downhill pyramidal cycle (101 011 111 110).

Lemma 5 If

$$\pi = (101, 011, 010, 110, \sigma) \tag{2}$$

is an arbitrary feasible sequence of L/U states, then the sequence

$$\pi' = (101, 011, 111, 110, \sigma) \tag{3}$$

is also feasible and the long run average cycle time of π' does not exceed the long run average cycle time of π.

Proof. The feasibility claim is obvious, so we concentrate on the comparison of cycle times. Informally, the dominance of π' over π can be seen as follows: In π', machine M_1 is loaded earlier and machine M_3 is unloaded later than in π, thus giving both machines more time for processing their respective parts. Since the robot travel time is identical in both sequences, the conclusion follows.

To establish this more formally, let us consider the execution of each sequence when starting from some arbitrary shop state in $e(0,1,1)$, say state $s = (0,1,1,0,0,p_2,\gamma,M_2)$ (notice that for any shop state in $e(0,1,1)$, the remaining time on M_2 is necessarily p_2 whereas the remaining time γ on M_3 is a priori unknown). We intend to show that, for any active schedule, the total time elapsed between s and the occurence of a first shop state in $l(1,1,0)$ is never larger for sequence π' than for sequence π. Moreover, if $(1,1,0,0,r_1,0,0,M_2)$ is the first state of $l(1,1,0)$ reached with π and $(1,1,0,0,s_1,0,0,M_2)$ is the first state of $l(1,1,0)$ reached with π', then $s_1 \leq r_1$. This is sufficient to establish that π' dominates π and thus to prove the lemma.

The trajectory of the shop in the state space is represented in Table 1 (for sequence π) and in Table 2 (for sequence π'). The tables are built as follows. The first column of each table traces the walk performed in G. For a row labelled $e(v)$ (resp. $l(v)$) the second and the third columns complete the description of the shop state $(v_1,v_2,v_3,w,r_1,r_2,r_3,pos)$ reached in $e(v)$ (resp. $l(v)$). N amely, the third column gives the position pos of the robot. Moreover, for each machine M_i $(i = 1,2,3)$,

• if machine M_i is not loaded, i.e. if $v_i = 0$, the corresponding entry in the second column is equal to 0;

• if machine M_i is loaded, i.e. if $v_i = 1$, the corresponding entry in the second column

is equal to the required processing time p_i minus the time elapsed since the machine has been loaded.
Therefore, a positive entry in the second column denotes remaining processing time and is necessarily equal to r_i for machine M_i. On the other hand, a negative entry indicates that the corresponding machine has finished processing.

Finally, the fourth column displays the transition time, i.e. the amount of time elapsed between the occurence of the shop state in the current row and the shop state in the previous row.

state	remaining processing time	*pos*	transition time
$e(0,1,1)$	$(0, p_2, \gamma)$	M_2	–
$l(0,1,1)$	$(0, p_2 - w, 0)$	M_3	$\max(\delta, \gamma) = w$
$e(0,1,0)$	$(0, p_2 - \delta - w, 0)$	M_4	δ
$l(0,1,0)$	$(0, p_2 - 5\delta - w, 0)$	M_0	4δ
$e(1,1,0)$	$(p_1, p_2 - 6\delta - w, 0)$	M_1	δ
$l(1,1,0)$	$(p_1 - \delta, 0, 0)$	M_2	$\max(\delta, p_2 - 6\delta - w)$

Table 1: Trajectory for sequence π.

state	remaining processing time	*pos*	transition time
$e(0,1,1)$	$(0, p_2, \gamma)$	M_2	–
$l(0,1,1)$	$(0, p_2 - 2\delta, \gamma - 2\delta)$	M_0	2δ
$e(1,1,1)$	$(p_1, p_2 - 3\delta, \gamma - 3\delta)$	M_1	δ
$l(1,1,1)$	$(p_1 - w', p_2 - 3\delta - w', 0)$	M_3	$\max(2\delta, \gamma - 3\delta) = w'$
$e(1,1,0)$	$(p_1 - \delta - w', p_2 - 4\delta - w', 0)$	M_4	δ
$l(1,1,0)$	$(p_1 - 3\delta - w', 0, 0)$	M_2	$\max(2\delta, p_2 - 4\delta - w')$

Table 2: Trajectory for sequence π'.

The total transition time between $e(0,1,1)$ and $l(1,1,0)$ is equal to $w+6\delta+\max(\delta, p_2-6\delta-w) = \max(w+7\delta, p_2)$ for the sequence π and is equal to $w'+4\delta+\max(2\delta, p_2-4\delta-w') = \max(w'+6\delta, p_2)$ for the sequence π'. Since $w'+6\delta \leq w+7\delta$, this transition time is no larger for π' than for π. Moreover, when $l(1,1,0)$ is reached, the remaining processing time on machine M_1 is also no larger for sequence π' than for sequence π. As a lready discussed, this is sufficient to establish the lemma. ■

Finally, we mention one last preliminary result.

Lemma 6 If $\max\{p_1, p_2, p_3\} \geq 8\delta$, then the sequence generated by the downhill cycle achieves minimum cycle time among all feasible robot move sequences. If $\max\{p_1, p_2, p_3\} < 8\delta$, then the cycle time of the downhill permutation is equal to 12δ.

Proof. See Sethi et al. [1992] or Crama and van de Klundert [1994] (note that 12δ is exactly the travel time required to execute the downhill cycle). ■

As a consequence of Lemma 6, we only need to prove the conjectures under the assumptions that $p_i < 8\delta$ for $i = 1, \ldots, 3$ and that the optimal cycle time is less than 12δ. These assumptions will be made in all subsequent developments .

In Subsections 4.2-4.5, we consider a fixed optimal k-unit cycle π and we proceed with the analysis of the four cases defined earlier.

4.2 Case 1

Lemma 7 If neither $e(0,0,1)$ nor $l(1,0,0)$ occurs in the optimal k-unit cycle π (that is, if Case 1 applies), then the downhill cycle (101 011 010 110) achieves the optimal cycle time.

Proof. If Case 1 applies, then the L/U state 100 never arises. Since 101 is the only other L/U state from which machine M_1 can be unloaded, the set $l(1,0,1)$ must be reached k times in the course of executing π. Consider now any two consecutive shop states in $l(1,0,1)$. When proceeding from the first to the next shop state, neither 100 nor 001 may arise. As can be checked from Figure 2, this only leaves two possible sequences of L/U states between these shop states: 101 011 010 110 101 or 101 011 111 110 101. The desired conclusion follows now from Lemma 5. ■

4.3 Case 2

The following observation can be made by inspection of Figure 2.

Observation 3 If $l(1,0,0)$ occurs in the optimal k-unit cycle π, but $e(0,0,1)$ does not (that is, if Case 2 applies), then the sequence of L/U states following 100 in π is of the type

$$100\ 010\ 110\ 101\ (011\ 111/010\ 110\ 101)^{k-1}. \tag{4}$$

Using this observation, we can now prove:

Lemma 8 If $l(1,0,0)$ occurs in the optimal k-unit cycle π but $e(0,0,1)$ does not (that is, if Case 2 applies), then the second pyramidal cycle (100 010 110 101) achieves the optimal cycle time.

Proof. First of all, there follows from Lemma 5 that, in the sequence (4), every occurence of the subsequence (101 011 010 110) can be disregarded and only the downhill subsequence $(101\ 011\ 111\ 110)^{k-1}$ needs to be considered.

We will prove the lemma by showing that the cycle time of π is at least k times the minimum of the cycle times of 100 010 110 101 and 101 011 111 110, namely the two pyramidal constituents of the sequence (4).

When $k = 1$ (that is, when π is the second pyramidal cycle), the scenario of Table 3 materializes. In this case, the cycle time of π amounts to $6\delta + p_3 + x + w$.

state	remaining processing time	*pos*	transition time
$l(1,0,0)$	(0,0,0)	M_1	–
$e(0,1,0)$	$(0, p_2, 0)$	M_2	δ
$l(0,1,0)$	$(0, p_2 - 2\delta, 0)$	M_0	2δ
$e(1,1,0)$	$(p_1, p_2 - 3\delta, 0)$	M_1	δ
$l(1,1,0)$	$(p_1 - x, 0, 0)$	M_2	$\max(\delta, p_2 - 3\delta) = x$
$e(1,0,1)$	$(p_1 - x - \delta, 0, p_3)$	M_3	δ
$l(1,0,1)$	$(p_1 - p_3 - x - \delta, 0, 0)$	M_3	p_3
$e(1,0,0)$	$(p_1 - p_3 - x - 2\delta, 0, 0)$	M_4	δ
$l(1,0,0)$	$(0,0,0)$	M_1	$\max(3\delta, p_1 - p_3 - x - 2\delta) = w$

Table 3: Case 2, $k = 1$.

Consider now the case where $k = 2$. This case is similar to the case $k = 1$ until a shop state in $e(1,0,1)$ is reached for the first time. Then, up to state $l(1,0,1)$, the scenario can be described as in Table 4.

After $l(1,0,1)$, the robot performs the downhill cycle 011 111 110 101 and comes back to a shop state in $e(1,0,1)$ with remaining processing times $(\alpha, 0, p_3)$, where $\alpha \leq p_1 - 6\delta$ (because the travel time of the robot after loading machine M_1 is equal to 6δ). The travel time between $l(1,0,1)$ and $e(1,0,1)$ amounts to 10δ. Thus, the sequence of shop states can be extended as shown in Table 4 (in the last line of this table, we have used the fact that $\alpha \leq p_1 - 6\delta$ and $p_1 \leq 8\delta$).

So, if $k = 2$, we conclude from Table 4 that the cycle time of π is at least

$$\frac{19\delta + p_3 + x + w'}{2},$$

where $x = \max(\delta, p_2 - 3\delta)$ and $w' = \max(2\delta, p_1 - x - \delta)$.

Now, trivially, $w' + \delta = \max(3\delta, p_1 - x) \geq \max(3\delta, p_1 - p_3 - x - 2\delta) = w$, and therefore

$$\begin{aligned} 19\delta + p_3 + x + w' &= 12\delta + 6\delta + p_3 + x + w' + \delta \\ &\geq 12\delta + (6\delta + p_3 + x + w). \end{aligned}$$

state	remaining processing time	*pos*	transition time
$l(1,0,0)$	(0,0,0)	M_1	–
$e(0,1,0)$	$(0,p_2,0)$	M_2	δ
$l(0,1,0)$	$(0,p_2-2\delta,0)$	M_0	2δ
$e(1,1,0)$	$(p_1,p_2-3\delta,0)$	M_1	δ
$l(1,1,0)$	$(p_1-x,0,0)$	M_2	$\max(\delta,p_2-3\delta)=x$
$e(1,0,1)$	$(p_1-x-\delta,0,p_3)$	M_3	δ
$l(1,0,1)$	$(0,0,p_3-w')$	M_1	$\max(2\delta,p_1-x-\delta)=w'$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$e(1,0,1)$	$(\alpha,0,p_3)$	M_3	at least 10δ since $l(1,0,1)$
$l(1,0,1)$	$(\alpha-p_3,0,0)$	M_3	p_3
$e(1,0,0)$	$(\alpha-p_3-\delta,0,0)$	M_4	δ
$l(1,0,0)$	$(0,0,0)$	M_1	$\max(3\delta,\alpha-p_3-\delta)=3\delta$

Table 4: Case 2, $k=2$.

This implies that, if the sequence (4) obtained for $k=2$ is optimal, then so are the sequence obtained for $k=1$ and the downhill sequence.

Finally, the above conclusion applies again when $k>2$, since each additional execution of the downhill sequence requires time at least 12δ. ■

4.4 Case 3

By inspection of Figure 2, we get:

Observation 4 If $e(0,0,1)$ occurs in the optimal k-unit cycle π, but $l(1,0,0)$ does not (that is, if Case 3 applies), then the sequence of L/U states following 001 in π is of the type

$$001\ (101\ 011\ 111/010\ 110)^{k-1}\ 101\ 011\ 010. \tag{5}$$

We rely on this observation to prove:

Lemma 9 If $e(0,0,1)$ occurs in the optimal k-unit cycle π but $l(1,0,0)$ does not (that is, if Case 3 applies), then the third pyramidal cycle (001 101 011 010) achieves the optimal cycle time.

Proof. The proof is very similar to the proof of Lemma 8. First of all, Lemma 5 implies again that the sequence displayed in equation (5) can be assumed to contain only the downhill subsequence (101 011 111 110) and not the sequence (101 011 010 110).

In case $k = 1$, namely when π is the third pyramidal cycle, then the scenario of Table 5 materializes and the cycle time amounts to $7\delta + p_1 + w + v$.

state	remaining processing time	*pos*	transition time
$e(0,0,1)$	$(0,0,p_3)$	M_3	–
$l(0,0,1)$	$(0,0,p_3 - 3\delta)$	M_0	3δ
$e(1,0,1)$	$(p_1,0,p_3 - 4\delta)$	M_1	δ
$l(1,0,1)$	$(0,0,p_3 - 4\delta - p_1)$	M_1	p_1
$e(0,1,1)$	$(0,p_2,p_3 - 5\delta - p_1)$	M_2	δ
$l(0,1,1)$	$(0,p_2 - w,0)$	M_3	$\max(\delta, p_3 - p_1 - 5\delta) = w$
$e(0,1,0)$	$(0,p_2 - w - \delta,0)$	M_4	δ
$l(0,1,0)$	$(0,0,0)$	M_2	$\max(2\delta, p_2 - w - \delta) = v$
$e(0,0,1)$	$(0,0,p_3)$	M_3	δ

Table 5: Case 3, $k = 1$.

Consider now the case where $k = 2$. This case is identical to the case $k = 1$ until a shop state in $e(0,1,1)$ is reached for the first time. Then, the scenario can be described as in Table 6: after leaving state $e(0,1,1)$, the shop proceeds to state $e(1,0,1)$ according to the downhill cycle. At this point, let α be the remaining processing time of machine M_1. It must be the case that $\alpha \leq p_1 - 6\delta$ since the travel time of the robot after loading M_1 is exactly 6δ. Moreover, the remaining processing times on M_2 and M_3 are respectively equal to $r_2 = 0$ and $r_3 = p_3$, since the robot just left $l(1,1,0)$. From Table 6, we deduce that the cycle time of π amounts to at least

$$\frac{19\delta + p_1 + w' + v'}{2}.$$

We now observe that, by definition of the quantities involved (see Tables 5 and 6), $w' \geq w$, $w + v = \max(2\delta + w, p_2 - \delta)$ and $w' + v' = \max(2\delta + w', p_2 - \delta)$. Therefore, $w' + v' \geq w + v$ and

$$\begin{aligned} 19\delta + p_1 + w' + v' &\geq 19\delta + p_1 + w + v \\ &= 12\delta + (7\delta + p_1 + w + v). \end{aligned}$$

The latter inequality implies that, if π is optimal, then so are the third pyramidal cycle ($k = 1$) and the downhill cycle.

When $k > 2$, the same conclusion applies (see the proof of Lemma 8) since each additional repetition of the downhill cycle requires time at least 12δ. ■

state	remaining processing time	*pos*	transition time
$e(0,0,1)$	$(0,0,p_3)$	M_3	–
$l(0,0,1)$	$(0,0,p_3-3\delta)$	M_0	3δ
$e(1,0,1)$	$(p_1,0,p_3-4\delta)$	M_1	δ
$l(1,0,1)$	$(0,0,p_3-4\delta-p_1)$	M_1	p_1
$e(0,1,1)$	$(0,p_2,p_3-5\delta-p_1)$	M_2	δ
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$e(1,0,1)$	$(\alpha,0,p_3)$	M_3	at least 9δ since $e(0,1,1)$
$l(1,0,1)$	$(0,0,p_3-2\delta)$	M_1	$\max(2\delta,\alpha)=2\delta$
$e(0,1,1)$	$(0,p_2,p_3-3\delta)$	M_2	δ
$l(0,1,1)$	$(0,p_2-w',0)$	M_3	$\max(\delta,p_3-3\delta)=w'$
$e(0,1,0)$	$(0,p_2-w'-\delta,0)$	M_4	δ
$l(0,1,0)$	$(0,0,0)$	M_2	$\max(2\delta,p_2-w'-\delta)=v'$
$e(0,0,1)$	$(0,0,p_3)$	M_3	δ

Table 6: Case 3, $k = 2$.

4.5 Case 4

By inspection of Figure 2, we deduce:

Observation 5 If both $e(1,0,0)$ and $l(0,0,1)$ occur in the optimal k-unit cycle π (that is, if Case 4 applies), then the sequence of L/U states occuring between 100 and 001 is either of the type 100 010 001 or of the type 100 010 110 101 (011 111/010 110 101)^a 011 010 001 for some $a \in \mathbb{N}$. Moreover, the sequence of L/U states occuring between 001 and 100 in π is either of the type 001 000 100 or of the type 001 101 (011 111/010 110 101)^b 100 for some $b \in \mathbb{N}$.

Lemma 10 If both $l(0,0,1)$ and $e(1,0,0)$ occur in the optimal cycle k-unit π (that is, if Case 4 applies), then one of the pyramidal cycles achieves the optimal cycle time.

Proof. From Observation 5 and from Lemmas 3-5, we deduce that π must be of one of the following four types:

100 010 001 000 (6)

$$100\ 010\ 110\ 101\ (011\ 111\ 110\ ;101)^{k-2}\ 011\ 010\ 001\ 000 \tag{7}$$

$$100\ 010\ 001\ 101\ (011\ 111\ 110\ 101)^{k-1} \tag{8}$$

$$100\ 010\ 110\ 101\ (011\ 111\ 110\ 101)^{a}\ 011\ 010\ 001\ 101\ (011\ 111\ 110\ 101)^{b} \tag{9}$$

where $a, b \in \mathbb{N}$ and $a+b = k-2$. Observe that the sequence (6) corresponds to the uphill cycle and that its cycle time is equal to $8\delta + p_1 + p_2 + p_3$.

Suppose next that $k = 2$ and that π is the sequence (7). This sequence is described in Table 7. From this table, we see that the total execution time of π amounts to

$$14\delta + p_1 + p_3 + w + x + y + z, \tag{10}$$

which is at least $20\delta + p_1 + p_3$. It must be the case that $p_1 \leq 4\delta$ and $p_3 \leq 4\delta$, otherwise the downhill sequence would have smaller cycle time than π. Thus, $x = 2\delta$ and $y = \delta$ and the total execution time of π is equal to

$$17\delta + p_1 + p_3 + w + z = 8\delta + p_1 + \max(2\delta, p_2 - 2\delta) + 9\delta + p_3 + \max(\delta, p_2 - 3\delta). \tag{11}$$

From the proofs of Lemma 8 and Lemma 9 (see Tables 3 and 5 respectively), there follows that the right-hand side of (11) is the sum of the cycle times of the second and third pyramidal cycles. Therefore, these two cycles must be op timal when π is.

Suppose next that π is the sequence (7) obtained when $k = 3$. This sequence is described in Table 8. Its total execution time is equal to $26\delta + p_1 + p_3 + w + x + y + z$, that is 12δ plus the execution time of the sequence (7) obtained when $k = 2$ (see (10)). Therefore, the downhill cycle, the second pyramidal cycle and the third pyramidal cycle are all optimal when π is.

When π is given by (7) and $k > 3$, the analysis is similar, since each additional execution of the downhill schedule requires 12δ time units. This takes entirely care of the case where π is the form (7).

Consider next the case where π corresponds to the sequence (8). If $k = 1$, then the travel time required by π is at least 12δ, as is easily checked (see also Theorem 4 in Sethi et al. [1992]). Hence, π is dominated by the downhill cycle. If $k = 2$, then the execution of π requires 20δ time units for robot moves only, plus p_2 time units for processing a part on machine M_2 between states 010 and 001, p_1 time units for processing on machine M_1 between states 101 and 011, and p_3 time units for processing on machine M_3 between states 101 and 100. Thus, in total, t he execution time of π is at least $20\delta + p_1 + p_2 + p_3$, and π is dominated by the downhill cycle (whose cycle time is 12δ) and by the uphill cycle (whose cycle time is $8\delta + p_1 + p_2 + p_3$).

Increasing k by one in (8) adds 12δ to the execution time of π and hence cannot reduce the cycle time.

Finally, let us consider the case where π is of the form (9). If $a = b = 0$, then the total travel time is already 24δ and if $a = 1, b = 0$, then it is 36δ. So, in both cases, π

state	remaining processing time	pos	transition time
$e(1,0,0)$	$(p_1,0,0)$	M_1	–
$l(1,0,0)$	$(0,0,0)$	M_1	p_1
$e(0,1,0)$	$(0,p_2,0)$	M_2	δ
$l(0,1,0)$	$(0,p_2-2\delta,0)$	M_0	2δ
$e(1,1,0)$	$(p_1,p_2-3\delta,0)$	M_1	δ
$l(1,1,0)$	$(p_1-w,0,0)$	M_2	$\max(\delta,p_2-3\delta)=w$
$e(1,0,1)$	$(p_1-\delta-w,0,p_3)$	M_3	δ
$l(1,0,1)$	$(0,0,p_3-x)$	M_1	$\max(2\delta,p_1-\delta-w)=x$
$e(0,1,1)$	$(0,p_2,p_3-\delta-x)$	M_2	δ
$l(0,1,1)$	$(0,p_2-y,0)$	M_3	$\max(\delta,p_3-\delta-x)=y$
$e(0,1,0)$	$(0,p_2-\delta-y,0)$	M_4	δ
$l(0,1,0)$	$(0,0,0)$	M_2	$\max(2\delta,p_2-\delta-y)=z$
$e(0,0,1)$	$(0,0,p_3)$	M_3	δ
$l(0,0,1)$	$(0,0,0)$	M_3	p_3
$e(0,0,0)$	$(0,0,0)$	M_4	δ
$l(0,0,0)$	$(0,0,0)$	M_0	4δ
$e(1,0,0)$	$(p_1,0,0)$	M_1	δ

Table 7: Case 4, $k=2$.

is dominated by the downhill cycle. If $a=0$ and $b=1$, consider the trajectory of shop states, starting from the (unique) sh op state in $l(1,0,0)$. Notice that π is identical to the sequence (7) obtained for $k=2$ up to the L/U state 101. Therefore, refering to Table 7, we deduce that $8\delta+w+x+y+z$ time units elapse between the first shop state in $l(1,0,0)$ and the first state in $e(0,0,1)$. From there on, the travel time of the robot amounts to 18δ until the shop state returns to $l(1,0,0)$. Moreover, the robot must wait for p_1 time units at machine M_1 between states 101 and 011, and p_3 time units at machine M_3 between states 101 and 100. Putting all these elements together, we see that a complete execution of π requires at least time $26\delta+p_1+p_3+w+x+y+z$, that is 12δ plus the execution t! ime of the sequence (7) obtained for $k=2$ (see expression (10)). There follows again that π is dominated by the downhill cycle as well as by the second and third pyramidal cycles.

Each additional execution of the downhill sequence (which arises when increasing a or b by 1) requires at least 12δ time units and cannot reduce the cycle time. ■

As a side-remark, we notice that, if π is chosen so that k is as small as possible among all optimal k-unit cyles, then the statement of Lemma 10 can be rephrased more precisely: namely, in that case, the reader can easily verify that the uphill pyramidal cycle

(6) achieves the optimal cycle time. In this sense, Lemmas 7-10 establish a one-to-one correspondence between the four cases described in Section 4.1 and the four pyramidal cycles.

state	remaining processing time	pos	transition time
$e(1,0,0)$	$(p_1,0,0)$	M_1	–
$l(1,0,0)$	$(0,0,0)$	M_1	p_1
$e(0,1,0)$	$(0,p_2,0)$	M_2	δ
$l(0,1,0)$	$(0,p_2-2\delta,0)$	M_0	2δ
$e(1,1,0)$	$(p_1,p_2-3\delta,0)$	M_1	δ
$l(1,1,0)$	$(p_1-w,0,0)$	M_2	$\max(\delta,p_2-3\delta)=w$
$e(1,0,1)$	$(p_1-\delta-w,0,p_3)$	M_3	δ
$l(1,0,1)$	$(0,0,p_3-x)$	M_1	$\max(2\delta,p_1-\delta-w)=x$
$e(0,1,1)$	$(0,p_2,p_3-x-\delta)$	M_2	δ
$l(0,1,1)$	$(0,p_2-2\delta,p_3-x-3\delta)$	M_0	2δ
$e(1,1,1)$	$(p_1,p_2-3\delta,p_3-x-4\delta)$	M_1	δ
$l(1,1,1)$	$(p_1-2\delta,p_2-5\delta,0)$	M_3	$\max(2\delta,p_3-x-4\delta)=2\delta$
$e(1,1,0)$	$(p_1-3\delta,p_2-6\delta,0)$	M_4	δ
$l(1,1,0)$	$(p_1-5\delta,0,0)$	M_2	$\max(2\delta,p_3-6\delta)=2\delta$
$e(1,0,1)$	$(p_1-6\delta,0,p_3)$	M_3	δ
$l(1,0,1)$	$(0,0,p_3-2\delta)$	M_1	$\max(2\delta,p_1-6\delta)=2\delta$
$e(0,1,1)$	$(0,p_2,p_3-3\delta)$	M_2	δ
$l(0,1,1)$	$(0,p_2-y,0)$	M_3	$\max(\delta,p_3-3\delta)=y$
$e(0,1,0)$	$(0,p_2-\delta-y,0)$	M_4	δ
$l(0,1,0)$	$(0,0,0)$	M_2	$\max(2\delta,p_2-\delta-y)=z$
$e(0,0,1)$	$(0,0,p_3)$	M_3	δ
$l(0,0,1)$	$(0,0,0)$	M_3	p_3
$e(0,0,0)$	$(0,0,0)$	M_4	δ
$l(0,0,0)$	$(0,0,0)$	M_0	4δ
$e(1,0,0)$	$(p_1,0,0)$	M_1	δ

Table 8: Case 4, $k=3$.

4.6 Putting the pieces together ...

We are finally ready to establish the validity of Conjecture 2.

Theorem 1 In a 3-machine no-buffer robotic flow shop with identical parts, there is a pyramidal 1-unit cycle that minimizes the long run average cycle time over the set of k-unit cycles, for all $k \in \mathbb{N}$.

Proof. The theorem is a consequence of Lemmas 7-10, which cover all possible cases. ■

We next use Theorem 1 to establish the validity of Conjecture 1:

Theorem 2 In a 3-machine no-buffer robotic flow shop with identical parts, the minimum long run average cycle time is achieved by a pyramidal 1-unit cycle.

Proof. Consider an instance of the robotic flow shop problem and denote by c^* the minimum cycle time achieved by pyramidal cycles. Consider an arbitrary feasible sequence of L/U states, say π (or, equivalently, the corresponding sequence of activities), where π is possibly non repetitive. Let S be the active schedule associated with π and let $c(\pi)$ be the cycle tim e of S. We want to show that $c(\pi) \geq c^*$.

If π does not contain the state 100, then Lemma 7 and Lemma 9 directly imply the claim. Thus, we may assume without loss of generality that 100 occurs infinitely many times and that 100 is the first L/U state in π. For each $t \in \mathbb{N}$, we are now going to introduce a k-cycle π_t, where $k = t$ or $k = t + 1$, such that $\pi_1^\infty, \pi_2^\infty, \ldots$ 'approximate' π more and more closely. More precisely, fix $t \in \mathbb{N}$, denote by $v = v(t)$ the L/U state of the shop just before the t-th execution of activity A_3 and by $w = w(t)$ the L/U state just after the t-th execution of activity A_3. Notice that (v, w) is one of the four pairs of states (101,100), (001,000), (111,110) or (011,010). We denote by (σ, v, w) the initial segment of π ending with (v, w), i.e. the finite sequence which is identical to π up to w. With these notations, we let π_t be the finite sequence

$$(\sigma, v, w) \text{ if } (v, w) = (101, 100), \tag{12}$$
$$(\sigma, v, w, 100) \text{ if } (v, w) = (001, 000), \tag{13}$$
$$(\sigma, v, w, 101, 100) \text{ if } (v, w) = (111, 110), \tag{14}$$
$$(\sigma, v, w, 001, 000, 100) \text{ if } (v, w) = (011, 010). \tag{15}$$

In each case, the idea is simply to quickly return the shop to the L/U state 100 after having performed (σ, v, w), where 'quickly' means: in constant time.

Clearly, the sequence π_t^∞ generated by π_t is feasible, so that π_t is a t-unit cycle in cases (12)-(13) and a $(t + 1)$-unit cycle in cases (14)-(15).

We claim that the following upper bound holds for the cycle time of π_t:

$$c(\pi_t) \leq \frac{S(A_3, t) + 10\delta + p_1 + p_2 + p_3}{t}. \tag{16}$$

Let us verify this claim for case (15) (the other cases are similar). We use Observation 2 (see Section 4.1). Since π_t starts and ends with 100, $c(\pi_t)$ is the time elapsed between

the first and last occurence of $l(1,0,0)$. Moreover, since π_t is identical to π up to states (v,w), we get:

$$c(\pi_t) \leq \frac{S(A_3,t)+r+p_1}{t+1},$$

where r is the time required to perform the sequence of activities A_3, A_2, A_3, A_0 associated to the sequence of L/U states 011 010 001 000 100 and p_1 is the waiting time between the end of A_0 and the start of the next activity (which is necessarily A_1). One easily computes that $r \leq 10\delta + p_2 + p_3$ and inequality (16) follows.

On the other hand, since π_t is finite, we know by Theorem 1 that $c(\pi_t) \geq c^*$. Together with (16), this implies that

$$\frac{S(A_3,t)}{t} \geq c^* - \frac{10\delta + p_1 + p_2 + p_3}{t}. \tag{17}$$

By Definition 7, we now obtain immediately

$$c(\pi) = \limsup_{t\to\infty} \frac{S(A_3,t)}{t} \geq c^*$$

and the proof is complete. ■

5 Generalizations and further research

A straightforward but rather tedious extension of the arguments presented in Section 4 shows that all our results go through without the assumptions that $\delta_{i,i+1} = \delta$ and $\epsilon_i = 0$ for $i = 0, \ldots, 4$ (see beginning of Section 3).

A much more challenging generalization would be to extend our results to m-machine robotic shops for m larger than three. We conjecture that the conclusion of Theorem 2 remains valid for an arbitrary number of machines, but we are presently unable to establish this claim.

Acknowledgements. The first author gratefully acknowledges partial support by the Office of Naval Research (grants N00014-92-J1375 and N00014-92-J4083) and by NATO (grant CRG 931531).

References

C.R. Asfahl, *Robots and Manufacturing Automation*, John Wiley and Sons, New York, 1985.

Y. Crama, Combinatorial models for production scheduling in automated manufacturing systems, *14th European Conference on Operational Research, Semi-plenary Papers*, pp. 237-259 (1995). To appear in *European Journal of Operational Research.*

Y. Crama and J. van de Klundert, Cyclic scheduling of identical parts in a robotic cell, 1994, to appear in *Operations Research.*

G. Finke, C. Gueguen and N. Brauner, Robotic cells with buffer space, *Proceedings ECCO IX,* (R. O'Connor and P. Magee, eds.), Dublin, Ireland, 1996.

P.C. Gilmore and R.E. Gomory, Sequencing a one state-variable machine: A solvable case of the traveling salesman problem, *Operations Research* 12 (1964) 655-679.

N.G. Hall and C. Sriskandarajah, A survey of machine scheduling problems with blocking and no-wait in process, *Operations Research* 44 (1996) 510-525.

N.G. Hall, H. Kamoun and C. Sriskandarajah, Scheduling in robotic cells: Classification, two and three machine cells, 1993, to appear in *Operations Research.*

A. Hertz, private communication, 1995.

V. Kats, private communication, 1995.

L. Lei, private communication, 1995.

R.W. Lieberman and I.B. Turksen, Crane scheduling problems, *AIIE Transactions* 13 (1981) 304-311.

S.T. McCormick and U.S. Rao, Some complexity results in cyclic scheduling, *Mathematical and Computer Modelling* 20 (1994) 107-122.

C.H. Papadimitriou and P.C. Kanellakis, Flowshop scheduling with limited temporary storage, *Journal of the ACM* 27 (1980) 533-549.

C. Sriskandarajah, N.G. Hall, H. Kamoun and H. Wan, Scheduling large robotic cells, Working Paper, College of Business, The Ohio State University, 1994.

S.P. Sethi, C. Sriskandarajah, G. Sorger, J. Blazewicz and W. Kubiak, Sequencing of parts

and robot moves in a robotic cell, *The International Journal of Flexible Manufacturing Systems* 4 (1992) 331-358.

J. van de Klundert, *Scheduling Problems in Automated Manufacturing*, Doctoral Dissertation, Maastricht University, 1996.