Cyclic scheduling in robotic flowshops

Y. Crama [1]
V. Kats [2]
J. van de Klundert [3]
E. Levner [4]

October 15, 1998

[1]Ecole d'Administration des Affaires, Université de Liège, Boulevard du Rectorat 7 (B31), 4000 Liège, Belgium. Email: Y.Crama@ulg.ac.be

[2]Department of Industrial Engineering and Management, Ben Gurion University of the Negev, POB 653, Beer-Sheva 84105, Israel. Email: VKats@bgumail.bgu.ac.il

[3]Department of Mathematics, Faculty of General Sciences, University of Maastricht, 6200 MD Maastricht, The Netherlands. Email: J.vandeKlundert@math.unimaas.nl

[4]Department of Computer Systems, Center for Technological Education, Holon 58102, Israel. Email: msegen@pluto.mscc.huji.ac.il

Cyclic scheduling in robotic flowshops

Abstract: Fully automated production cells consisting of flexible machines and a material handling robot have become commonplace in contemporary manufacturing systems. Much research on scheduling problems arising in such cells, in particular, in flowshop-like production cells has been reported recently. Although there are many differences between the models, they all explicitly incorporate the interaction between the materials handling and the classical job processing decisions, since this interaction determines the efficiency of the cell. This paper surveys cyclic scheduling problems in robotic flowshops, models for such problems, and the complexity of solving these problems, thereby bringing together several streams of research that have by and large ignored one another, and describing and establishing links with other scheduling problems and combinatorial topics.

1 Introduction

Over the last couple of decades, the automation of production technology, in combination with advances in production management, has drastically changed the equipment used by manufacturing companies as well as the issues arising in production planning and control. Together, these changes have led to an enormous increase in efficiency and flexibility, so that progress in terms of automation has become a necessity to survive global competition. As a consequence, highly automated or even unmanned manufacturing systems have become commonplace in several industrial sectors, especially in mechanical engineering and electronics.

Typically, automated manufacturing systems involve intelligent, flexible, machines, that can be programmed to produce a variety of parts with little or no setup times. Often, these flexible machines are grouped into cells in such a way that the entire production of each part can be performed within one of the cells. One of the benefits of machine pooling is the reduction in material handling activities that it entails. Within a cell, material handling is usually performed by one (or a few) robots or automatic guided vehicles (AGVs). When this is the case, the performance of the cell becomes highly dependent on the interaction between the automated material handling device(s) and the machines (see e.g. Asfahl [1985] pp. 272-281).

The relatively small number of machines and material handling device(s) in flexible cells, as well as their high degree of automation, make them an ideal environment for automated production scheduling. As a matter of fact, in several industrial applications, the use of advanced planning software has been reported to improve substantially the performance of the cells. Classical scheduling models however, as they have been developed until the late seventies, appear to be unsuitable to incorporate the most important characteristics of flexible manufacturing cells, such as the interaction between the material handling system and the machines. Hence, a diverse lot of new and challenging scheduling problems has recently appeared in the literature. In this survey, we attempt to take a systematic view on those problems that specifically deal with the aforementioned interaction between the material handling device(s) and the machines. In particular, we focus on a class of problems known as *robotic flowshop scheduling problems.*

In spite of the fact that most of the research on this topic is quite recent, it is interesting to remark that several important results date back more than three decades (Suprunenko et al. [1962], Aizenshtat [1963a, 1963b] and Tanayev [1964]) and that, to date, a wide variety of robotic cell scheduling problems have already been investigated. In this survey, we classify and overview models, problems, algorithms and complexity results mentioned in the literature and discuss their interrelationships. Our emphasis is on constructive and exact results in the realm of deterministic scheduling, rather than empirical or simulations studies. Our contribution lies in establishing and discussing the commonalities and differences between papers that largely ignore one another, and in showing their relationship with other combinatorial problems. As such, this survey on robotic scheduling is related and complementary to the surveys of Serafini and Ukovich [1989] on periodic scheduling, Blazewicz and Finke [1994] on resource constrained scheduling in manufacturing systems, Hanen and Munier [1995] on cyclic scheduling for parallel processors, Hall and Sriskandarajah [1996] on flowshop scheduling, Crama [1997] on combinatorial problems in automated manufacturing and Hall, Kamoun and Sriskandarajah [1997] on scheduling in small-scale robotic cells.

In Section 2, we present various robotic cell layouts and scheduling problems as they

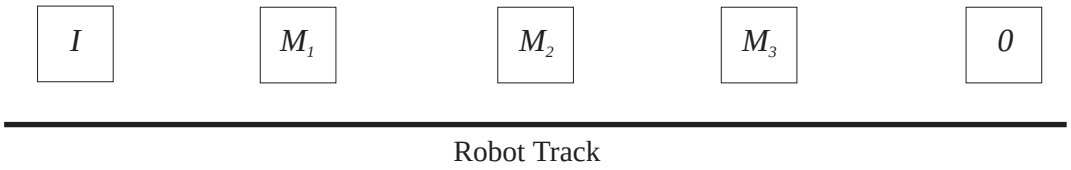

Figure 1: A 3-machine robotic cell (line layout)

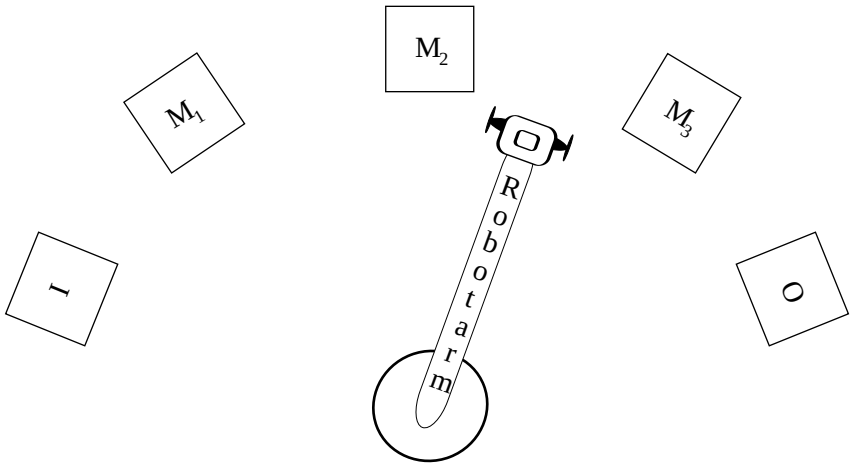

Figure 2: A 3-machine robotic cell (line layout)

have been described in the literature. This section does not attempt to give an exhaustive overview of the area of robotic cell scheduling. As a matter of fact, as this research area is moving and expanding rapidly, we do not even claim to give a complete overview of the topics that we study at length in subsequent sections. We only pay attention to models and issues which we believe to be particularly important and interesting. Our main focus is on cyclic robotic flowshop scheduling problems without buffers (see Figures 1 and 2). We note that this emphasis on bufferless systems is consistent with modern production philosophies such as Just In Time or lean manufacturing.

A *robotic flowshop* consists of m machines $M_1, \ldots, M_m$, an input station M_0, an output station M_{m+1} and one (or several) robot(s). The robots perform all material handling operations in the cell, i.e. the transportation of parts between the machines and the stations, as well as the loading and unloading of parts onto and from the machines and stations. All parts are initially available at the input station and must be sequentially processed on M_1, $M_2, \ldots, M_m$, until they are finally unloaded from M_m and delivered at the output station.

A most general optimization problem for robotic flowshops asks to specify a *sequence of robot moves* and a *part input sequence* (i.e. the order in which the parts are to be taken from the input station) as well as a *schedule* for the operations associated to these sequences, so as to maximize a predetermined production performance measure (for instance, the throughput rate of the flowshop). In Section 3, we discuss sequences and schedules in more detail.

In particular, we describe some of their desirable properties and their interrelationships. Section 4 is devoted to a survey of the complexity of several variants of this general flowshop scheduling problem. A number of variants, in which one of the two sequencing problems has vanished or is treated as given, turn out to be particulary interesting. Section 5 deals with problems that arise when both sequences are treated as given. More precisely, we discuss the problem of computing the optimal cycle time of a production plan in which both the robot move sequence and the part input sequence are given. We focus on the best available algorithms and, in some occasions, we point out improvements and/or generalizations of previously published results. In Section 6, we comment on various other aspects of robotic cell scheduling problems and we suggest a number of new research directions.

2 Robotic flowshops: modeling the system

In this section, we provide a general description of (a variety of) robotic cell models. The emphasis lies here mostly on the *data* of the problems, i.e. on the way in which real-world characteristics of robotic flowshops are captured by various models. In the next section, by contrast, we shall focus on the description of the *output*, i.e. on the properties which can be imposed on *solutions* of the problem. We start our discussion with the case of bufferless robotic flowshops.

2.1 Basic ingredients

Let us first collect the basic constituents of a bufferless robotic flowshop.

1. There are m machines denoted $M_1, \ldots, M_m$.
2. There is an input station denoted M_0 and an output station denoted M_{m+1}.
3. There is a set of parts J. Each part has prespecified processing requirements on each machine M_i, for $i = 1, \ldots, m$ (see below for a more precise statement of this assumption). The parts are initially available at the input station, must be processed on every machine in increasing order of machine indices and must finally be delivered at the output station.
4. There is a robot that performs the transportation of parts between the machines. The loading and unloading of parts onto or from the machines also requires the attendance of the robot.
5. The input and the output station have infinite capacity. There are no other buffers in the flowshop and the machines as well as the robot can only process one part at a time. In particular, if a part is between the input and output station, then it is either on a machine or being carried by the robot.

In general terms, the robotic flowshop scheduling problem is to determine an ordering of the parts at the input station (a *part input sequence*), a sequence of the robot activities (a *robot move sequence*) and the start times and finish times of these activities (a *schedule*) so as to optimize production performance. We shall return to these issues in the next section and we shall see that, in many cases, an optimal schedule can be efficiently computed once

the part input sequence and the robot move sequence are known. For this reason, the robotic cell scheduling problem can often be seen as a *double sequencing problem.*

In the following subsections, we briefly discuss the distinguishing features of various types of flowshops.

2.2 Processing requirements

In very general form, the processing requirements of part j on machine M_i can be modeled by means of a *processing window* $[L_i^j, U_i^j]$. The interpretation is that the time spent by part j on machine M_i must be at least L_i^j and may not exceed U_i^j. Such processing requirements arise for instance quite naturally in manufacturing processes observed in the electronics industry. Here, printed circuit boards must be sucessively dipped into several chemical baths and the duration of each chemical bath may be neither too short nor too long. Other applications can be found in the food processing industry or in printed circuit board assembly, as well as in medical laboratories (see, among others, Livshits et al. [1974], Phillips and Unger [1976], Lei and Wang [1991], Hertz et al. [1996], Chu and Proth [1996]).

Processing windows are general enough to contain many other specifications of the processing requirements as special cases. For instance, the classical situation in which each part has a precisely defined processing time on each machine and can wait on the machine indefinitely long after it has been processed, arises by setting all upperbounds U_i^j to $+\infty$. Such problems have been investigated in several papers; see e.g. Sethi et al. [1992],Hall et al. [1997], Crama and Van de Klundert [1997a], Sriskandarajah et al. [1998], and the references therein. In the remainder of this survey we will refer to this case as *unbounded processing windows.* In the context of classical flowshop scheduling, analogous problems are known as problems *with blocking.* Also, several authors consider a *no-wait* version of our basic model, in which all parts must be unloaded as soon as they finish processing. This can be modeled by setting $L_i^j = U_i^j$, see e.g. Aizenshtat[1963], Livshits et al.[1974], Kats [1982], Levner, Kats and Levit [1997]. We will refer to this case as *zero-width processing windows.*

In the same spirit as processing windows, production characteristics may also impose lower and upper limits on the duration of material handling activities (Livshits et al. [1974], Ng and Leung [1997]). For instance, in certain environments, the robot may have to load each part on M_i as soon as possible after unloading it from M_{i-1}. Such a restriction is referred to as *loaded-robot no-wait.* The alternative is, of course, that the robot may pause between unloading a part from M_{i-1} and loading it on M_i. In the literature, loaded-robot no-wait restrictions have only been considered in conjunction with zero-width processing windows, so that we will restrict our discussion to this case (although in practice, the two features need not be combined).

2.3 Time models for the robot

In order to completely define an instance of a robotic cell scheduling problem, we need to further specify the behavior of the robot. Several authors (see e.g. Kamoun et al. [1993], Hall et al. [1997], Sriskandarajah et al. [1998]) consider the duration of the load and unload activities to be machine dependent: their models assume that it takes time ϵ_i to load a part on machine M_i or to unload the part from this machine ($i = 0, \ldots, m+1$). Job dependent models have been investigated in Levner, Kogan and Maimon [1995] and Kogan and Levner

[1997]. Simpler models, in which all load and unload times are equal ($\epsilon_i = \epsilon$ or even $\epsilon_i = 0$) can for instance be found in Kise, Shioyama and Ibaraki [1991].

An important characteristic of the robot is its travel speed. We denote by δ_{ij} the time needed for the robot to travel from machine M_i to machine M_j. The travel times are often assumed to be symmetric ($\delta_{ij} = \delta_{ji}$ for all i, j) and to satisfy the triangle inequality (for $i < j < k, \delta_{ij} + \delta_{jk} \geq \delta_{ik}$). Alternatively, several authors have considered models in which the travel times can differ for loaded and unloaded robot moves (see e.g. Phillips and Unger [1976], Hertz et al. [1996], etc). Such variations in the robot time model may influence the problem complexity (see e.g. Levner, Kats and Levit [1997]). Simpler models, in which travel times are assumed to be negligible when the robot is unloaded, have been studied by Song et al. [1993] and Chu and Proth [1996].

In a flowshop, the machines are often laid out in line (see again Figures 1 and 2). When this is the case, the travel times can be assumed to be symmetric and to satisfy the following relation : for $i < j < k, \delta_{ij} + \delta_{jk} = \delta_{ik}$. In the sequel, we refer to this equality as the *triangle equality*. In case the triangle equality holds, a correction factor may be introduced to model the fact that the robot is carrying a part.

2.4 Number of machines and parts

Obviously, the problem size depends, among other factors, on the number of machines, parts and part types. The general robotic cell scheduling problem (to be introduced shortly) is hard to solve, so that special, more tractable cases of the problem have received much attention in the literature.

First of all, in many practical instances, the number of machines is relatively small. This justifies the interest in cells consisting of only two or three machines. More generally, regarding the number of machines as a constant, rather than as input data, yields interesting algorithmic and theoretical insights.

Similar remarks hold for the number of distinct part types to be processed. Actually, a fair deal of research has been devoted to problems in which all parts are identical, i.e. have the same processing requirements. Notice that in this case, the part input sequencing problem vanishes altogether.

More generally, the parts are often partitioned into a small number (say, Q) of *part types*, with the property that all parts of a same type possess exactly the same processing requirements. A *minimal part set* (MPS) is then described by a vector of the form $(n_1, n_2, \ldots, n_Q)$ where n_q indicates the number of parts of type q contained in this set ($1 \leq q \leq Q$) and $n_1, n_2, \ldots, n_Q$ are relatively prime (Hitz [1980]). The part set J consists of a number of copies of the MPS (possibly, infinitely many copies) to be produced repeatedly. Very little is known about the complexity of the problems arising in this setting when the number of part types is regarded as a constant (see e.g. Pinedo [1995], Lee and Posner [1997]).

2.5 Objective functions

The mainstream of research in robotic flowshop scheduling is devoted to only two classes of production performance measures. In order to introduce them, let us denote by S_n ($n = 1, 2, \ldots$) the *completion time* of the n-th part processed in a given schedule, that is the moment at which the n-th part is unloaded at the output station. Then, one class of models addresses

the case of a finite part set J with the objective of minimizing the *makespan* of the schedule, viz. $\max_{n=1,\ldots,|J|} S_n$ (see e.g. Kise [1991], Kise et al. [1991], Hertz et al. [1996], Chu and Proth [1996]). Another class of models assumes that J is infinite and attempts to minimize the *long run cycle time* of the schedule, viz. $\limsup_{n\to\infty} \frac{S_n}{n}$ (see e.g. Sethi et al. [1992]; notice that this objective is equivalent to maximizing the throughput rate).

2.6 Other types of robotic cells

Several interesting papers on robotic cell scheduling deal with variants of the bufferless robotic flowshop discussed above.

An important class of problems addresses robotic cells involving more than one robot. Such multi-robot cells are common in contemporary manufacturing systems. However, we have chosen not to treat them in great depth for two reasons. From the viewpoint of robot move sequencing, these problems seem to be only remotely related to the problems that we consider, and their complexity is as yet not very well understood. From the viewpoint of cycle time computation, on the other hand, these problems can usually be solved by straightforward generalizations of the techniques described in Section 5. Problems of this type are described in Karzanov and Livshitz [1978], Liebermand and Turksen [1981], Lei and Wang [1994], Kats and Levner [1998b] and the references therein. When dealing with the design of robotic cells, the number of robots is also sometimes treated as a decision variable (see for instance Lei, Armstrong and Gu [1993] or Kats and Levner [1998b]; see also Orlin [1982] for a related study).

Other authors have considered problems that optimize different objective functions. For instance, Song et al. [1993] and Jeng et al. [1993] try to minimize the sum of the completion times. Levner and Vlach [1997] consider a closely related problem in which the objective is to minimize some penalty function on the maximum lateness.

Several authors study robotic cells in which the machines are equipped with (input or output) buffers where the parts can wait until the machine or the robot becomes available. For instance, Park [1995] uses simulation to investigate the effectiveness of various dispatching rules in such an environment. King et al. [1993] propose a branch and bround algorithm for a similar problem. On the other hand, theoretical work has also been reported on questions that parallel those encountered in the previous subsections. For instance, Finke et al. [1996] identify optimal robot move sequences for a problem with single capacity buffers. Levner, Kogan and Maimon [1995] and Kogan and Levner [1997] consider the two-machine problem with infinite intermediate buffer, which can be solved using a polynomial-time Johnson-type algorithm (Johnson [1954]) when each machine is supplied with its own servicing robot. Kise [1991] shows that minimizing the makespan is NP-hard for single-robot flowshops since in this case the robot can be looked at as a third machine in the shop.

As a general rule, the additional freedom introduced by finite capacity buffers tends to complicate scheduling problems (see e.g. Papadimitriou and Kanellakis [1980]). Buffers may allow, for example, to consider non-permutation schedules, i.e. schedules in which parts are processed in different orders on different machines.

Finally, non flowshop variants of robotic cell problems have also been investigated. For instance, re-entrant flowshops have been studied by Kats and Levner [1997] and cells with parallel machines have been considered in Hall, Potts, and Sriskandarajah [1995], Glass et al. [1995], and Jeng et al. [1993]. Hertz et al. [1996], consider a re-entrant flowshop in which

some of the machines may handle more than one part at a time, and the time elapsed between the end of an activity and the beginning of another one may be bounded. Actually, a robot could theoretically be added to any shop layout, thus giving rise to further interesting models.

In order to better appreciate the practical relevance of the various robotic cell models and the associated scheduling problems, a survey of case studies in an industrial framework would be of much help, but currently appears to be lacking.

3 Robotic flowshops: sequencing and scheduling

From now on, we focus on a version of the bufferless robotic flowshop problem displaying the following features (some of which will be further restricted in subsequent discussions). The travel times δ_{ij} between machines ($i, j = 0, \ldots, m+1$) are symmetric and satisfy the triangle equalities. The loading and unloading time of machine M_i is equal to ϵ_i, for $i = 0, \ldots, m+1$. The processing requirements of part j on machine M_i are defined by a processing window $[L_i^j, U_i^j]$ ($j = 1, 2, \ldots; i = 1, \ldots, m$). The objective is to determine a schedule with minimum long run cycle time. More formally, the problem can be stated as follows:

Definition 1 *Robotic Flowshop Scheduling Problem* (RFS):

INPUT : δ_{ij}, ϵ_i, $[L_i^j, U_i^j]$ for $i = 0, \ldots, m+1$ and $j = 1, 2, \ldots$;

QUESTION : Find a part input sequence, a robot move sequence and a corresponding schedule with minimum long run cycle time.

This definition emphasizes that a solution of the RFS problem actually involves *sequencing* and *scheduling* issues. However, at this point, it is not at all clear how these issues relate to each other: how do we determine a schedule that is consistent with a given part input sequence and robot move sequence? how do we compute the corresponding long run cycle time? etc. Further, we have assumed so far that every robot move sequence and every part input sequence is allowed. In fact, from a practical point of view, this is usually far from convenient or realistic (especially when the number of parts is infinite!). This explains that most authors have found it desirable to impose restrictions on the type of solutions that are allowed (some of them will be examined below).

In the next subsections, we are going to discuss sequences, schedules and their special properties in more detail.

3.1 Robot move sequences

In the sequel, we always assume that we are free to specify the initial state (loaded or unloaded) of the machines (in particular, we do not assume that the cell is initially empty). This assumption is reasonable as long as we concentrate, as we do, on a long run performance measure.

Remember that the robot performs three kinds of operations: loading, unloading and transportation of parts. Since the robot can only handle one part at a time, it must necessarily unload machine M_i immediately before it loads machine M_{i+1}, for $i = 0, \ldots, m$. This observation motivates the following definition.

Definition 2 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* A_i, for $i = 0, \ldots, m$.

Additional robot movement is of course required between the loading of a machine, i.e. the completion of an activity, and the unloading of a next machine, i.e. the start of the subsequent activity. Nevertheless, the sequence of moves and tasks performed by the robot can always be described as a sequence of activities.

Definition 3 A (possibly infinite) sequence π of robot activities is called a *feasible robot move sequence* if, in the course of executing the sequence,

1. the robot is never required to unload an empty machine and
2. the robot is never required to load a loaded machine.

A more operational version of this definition can be stated as follows (Crama and Van de Klundert [1997]).

Proposition 1 A (possibly infinite) sequence of activities π is feasible if and only if

a. 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 in π, and

b. A_1 occurs exactly once between any two consecutive occurences of A_0 in π, and

c. A_{m-1} occurs exactly once between any two consecutive occurences of A_m in π.

Proof. See Appendix.

Example 1 *For a 3-machine robotic flowshop, the sequence*

$$A_3, A_2, A_1, A_0, A_3, A_2, A_1, A_3, A_2, A_3$$

is feasible.

As already pointed out, describing an optimal robot move sequence may very well turn out to be an untractable task, since such a sequence may – a priori – be infinitely long. It is therefore customary to restrict the solution set of the RFS problem analysis to those infinite sequences which arise by repeating a 'short' robot move sequences infinitely many times (although this may potentially result in suboptimal schedules).

If π is a finite sequence of robot activities, let us denote by π^∞ the sequence obtained by repeating π indefinitely. We say that π is *repeatable* if π^∞ is feasible. Of course, it is not the case that every sequence is repeatable: Example 1 provides a counter-example. As an easy consequence of Proposition 1, we obtain the following characterization of repeatable sequences.

Proposition 2 A finite sequence π is repeatable if and only if, when π is regarded as a cyclic sequence, each of A_{i-1} and A_{i+1} occurs exactly once between any two consecutive occurences of A_i for $i = 1, \ldots, m-1$.

Observe that, in particular, every activity $A_0, A_1, \ldots, A_m$ must occur the same number of times in a repeatable sequence. A repeatable sequence in which each activity appears k times is called a *k-unit cycle* (the terminology *k-unit sequence* would probably be preferable, but we abide here by an entrenched custom). When the robot has finished executing a k-unit cycle, exactly k parts have been produced (i.e., have been unloaded at the output station) and the flowshop has returned to its original state (since each machine has been loaded and unloaded exactly k times).

Much of the literature on robotic flowshop scheduling has focused on the computation of optimal 1-unit cycles (in spite of the fact that there are only very few cases in which this restriction is known not to cause an increase in the minimum possible long run cycle time). In view of Proposition 2, a 1-unit cycle is nothing but a permutation of the activities $A_0, A_1, \ldots, A_m$ (this has been observed by Lieberman and Turksen [1981] and Sethi et al. [1992]). Thus, for a 1-unit cycle, the order in which the robot activities are performed remains constant for all parts. We will frequently return to 1-unit cycles in subsequent sections.

3.2 Part input sequences

For the same reasons as in the case of robot move sequences, it can prove difficult to solve the RFS problem over the set of *all* possible sequences of parts. When the parts to be processed are described by a minimal part set $(n_1, n_2, \ldots, n_Q)$, it is therefore convenient (both from a theoretical and practical viewpoint) to restrict the search to those part input sequences which consist in indefinitely repeating the production of an MPS and to process each MPS in the same order. Thus, if $h = \sum_{q=1}^{Q} n_q$, then the part input sequence repeats after every h parts. This restriction, which is common in JIT manufacturing, has been adopted in all the investigations of which we are aware.

Example 2 *If the part set J consists of 100 parts of type A, 100 parts of type B and 200 parts of type C, then the MPS is $(1, 1, 2)$ and the complete part set can be processed by repeatedly processing the MPS in the sequence $ACBC$ (produce first a part of type A, then a part of type C, etc.).*

3.3 Schedules

Let us now turn our attention from sequences to schedules.

Definition 4 A *schedule* S is defined as a specification of starting times for each load and unload operation. More precisely, $S(l, i, t)$ (resp. $S(u, i, t)$) the time at which the t-th loading (resp. unloading) of machine M_i starts according to S $(i = 0, \ldots, m;\ t \in \mathbb{N})$.

In a feasible schedule, the start times $S(u, i, t)$ and $S(l, i, t)$ $(i = 0, \ldots, m;\ t \in \mathbb{N})$ can be ordered chronologically. For a feasible schedule S, let $\pi(S) = \pi$ be the *unique* feasible robot move sequence in which the activities occur in the chronogical order defined by S. We say that $\pi(S)$ is the sequence *implied by* S. Observe that the converse relation must be more

carefully defined: indeed, several schedules may very well imply the same sequence of moves. Therefore, we simply say that S *is a schedule for* a robot move sequence π if S implies π.

Let us now study conditions under which a schedule S is *feasible* for a given robot move sequence π and part input sequence. We must first ensure that, between loading and unloading of a machine, the machine has enough time for processing. Let j be the part loaded onto machine M_i in the t-th loading operation. If the t-th loading takes place before the t-th unloading in π, then :

$$S(u,i,t) - S(l,i,t) \geq L_i^j + \epsilon_i, \tag{1}$$

and

$$S(u,i,t) - S(l,i,t) \leq U_i^j + \epsilon_i. \tag{2}$$

Otherwise, if the t-th unloading takes place before the t-th loading, then :

$$S(u,i,t+1) - S(l,i,t) \geq L_i^j + \epsilon_i, \tag{3}$$

and

$$S(u,i,t+1) - S(l,i,t) \leq U_i^j + \epsilon_i. \tag{4}$$

The schedule S must allow the robot enough travel time between consecutive load/unload operations :

$$S(l,i+1,t) - S(u,i,t) \geq \epsilon_i + \delta_{i,i+1}. \tag{5}$$

Furthermore, if the t-th execution of A_k (directly) succeeds the s-th execution of A_i in π then

$$S(u,k,t) - S(l,i+1,s) \geq \epsilon_{i+1} + \delta_{i+1,k}. \tag{6}$$

If π is a feasible robot move sequence, conditions (1)-(6) are necessary and sufficient for the feasibility of any schedule S for π: the time intervals elapsing between loading and unloading do not violate the processing windows, because of (1)-(4), and the robot is allowed enough time for travelling between load and unload operations, because of (5)-(6). Moreover, since π is a feasible robot move sequence, the robot does not unload any empty machines, nor does it load any busy machine.

Let us now impose some additional structure on the schedules to be considered.

Definition 5 A schedule S is said to be *periodic* if there exists a constant $r \in \mathbb{N}$, called the *period* of S, and a constant $C_S \in \mathbb{R}$, called the *cycle time* of S, such that $S(l,i,t+r) - S(l,i,t) = r \times C_S$ and $S(u,i,t+r) - S(u,i,t) = r \times C_S$ for all $i = 0, 1, \ldots, m$ and all $t \in \mathbb{N}$. We say that S is *r-periodic* if S is periodic with period r.

Obviously, the long run cycle time of a periodic schedule S is equal to C_S. Hence, considering only periodic schedules provides a way to circumvent some of the difficulties encountered in the computation of the long run cycle time for arbitrary schedules. In particular, for part input sequences obtained by repetition of a same MPS (see previous subsection), it is quite natural (though not necessarily optimal) to restrict the attention to h-periodic schedules, where h is the number of parts in each MPS. In Section 5, we will discuss the complexity of computing a periodic schedule when the robot move sequence and the part input sequence

are fixed. For now, let us simply show how the system of inequalities (1)-(6) can be adapted in order to compute a feasible 1-periodic schedule associated to a given 1-unit cycle in the case where all parts are identical (this is the case $h = 1$ in the foregoing discussion; the more general problem of computing a h-periodic schedule associated with a given k-cycle could be similarly modelled).

Since all parts have identical processing requirements, we write the processing windows as $[L_i, U_i]$ for $i = 1, \ldots, m$. We denote by π the (known) 1-unit cycle, by S the (unknown) schedule and by C_S its (unknown) cycle time.

If A_{i-1} precedes A_i in π, then inequalities (1)-(2) become:

$$S(u, i, t) - S(l, i, t) \geq L_i + \epsilon_i, \tag{7}$$

$$S(u, i, t) - S(l, i, t) \leq U_i + \epsilon_i. \tag{8}$$

On the other hand, if A_i precedes A_{i-1} in π, then

$$S(u, i, t) - S(l, i, t) \geq L_i + \epsilon_i - C_S, \tag{9}$$

$$S(u, i, t) - S(l, i, t) \leq U_i + \epsilon_i - C_S. \tag{10}$$

The robot must again be allowed enough time to perform each activity :

$$S(l, i + 1, t) - S(u, i, t) \geq \epsilon_i + \delta_{i,i+1} \tag{11}$$

Furthermore, if A_k is the activity succeeding A_i in π, then

$$S(u, k, t) - S(l, i + 1, t) \geq \epsilon_{i+1} + \delta_{i+1,k}. \tag{12}$$

If A_i is the last activity in π and A_k is the first, then

$$S(u, k, t) - S(l, i + 1, t) \geq \epsilon_{i+1} + \delta_{i+1,k} - C_S. \tag{13}$$

Notice that (13) arises from (6) because π is repeated infinitely many times.

We will have several opportunities to return to this formulation in subsequent sections. At this point however, we briefly discuss the existence of l-periodic schedules for k-unit robot move sequences, where $k, l \in \mathbb{N}$. The above formulation contains two constraints for each of the m activities and $m + 1$ constraints for the robot moves between consecutive activities. Thus, in total the formulation contains $3m+1$ constraints. Likewise, it is possible to formulate a linear program consisting of $6m + 1$ constraints to compute the cycle time of a 2-unit cycle. According to Definition 5, the resulting schedule will be a 2-periodic schedule. If the 2-unit cycle consists of two repetitions of the same 1-unit cycle, we obtain in this way a 2-periodic schedule for a 1-unit cycle. In subsequent sections we return to the question whether such a 2-periodic schedule for a 1-unit cycle may yield lower cycle times than 1-periodic cycles for the same 1-unit cycle. Observe also that the reverse situation, namely a 1-periodic schedule for a 2-unit cycle, can only arise if the 2-unit cycle is obtained by repeating twice a same 1-unit cycle. Indeed, the robot move sequence implied by 1-periodic schedule necessarily repeats itself after the execution of some initial 1-unit cycle.

4 Optimization problems and models

In the current section, we discuss the complexity of, and algorithms for the robotic flowshop scheduling (RFS) problem when either the part input sequence or the robot move sequence is *not* fixed. To place the results in perspective, we start with a brief review of the complexity of traditional bufferless flowshop scheduling problems.

The best known result in bufferless flowshop scheduling is probably the algorithm proposed by Gilmore and Gomory [1964] for makespan minimization in the two-machine case. These authors showed that the two-machine problem can be modeled as a special kind of traveling salesman problem (TSP) and proved that this TSP can be solved in polynomial time. By contrast, the decision version of the three-machine bufferless flowshop problem is strongly NP-complete. As observed by McCormick and Rao [1994], this follows from a result of Papadimitriou and Kanellakis [1980], who proved that the decision version of the two-machine flowshop problem with a unit capacity buffer is strongly NP-complete.

The situation is quite similar for no-wait flowshop scheduling problems. In the two-machine case, there is no difference between no-wait and no-buffer and for three machines the decision version of the no-wait problem is strongly NP-complete (Rock [1984]). For any number of machines, no-wait flowshop scheduling problems can be modeled as TSPs. More details can be found in Levner [1969], Kamoun and Sriskandarajah [1993], and Hall and Sriskandarajah [1997], who discuss a variety of no-wait and no-buffer flowshop scheduling problems.

Finally, although this situation has not received much attention in the literature, let us consider the case where the processing requirements of a traditional no-buffer flowshop are expressed by means of processing windows. In the two-machine case, the problem can be solved by the Gilmore-Gomory algorithm, simply by setting the processing times equal to the lowerbounds and solving as a no-wait scheduling problem thereafter. On the other hand, since it contains the no-wait problem as a special case, the decision version of the problem with processing windows is strongly NP-complete for three or more machines.

All the complexity results mentioned above concern the makespan minimization objective. McCormick and Rao [1994] address the relation between makespan minimization and cycle time minimization for traditional flowshop scheduling problems. They establish that makespan minimization problems can be efficiently transformed into cycle time minimization problems. Thus, the latter model is more general than the former one. Striclty speaking however, their transformation assumes that the part input sequence must be the same for each MPS produced. This requirement, which is appealing from a theoretical as well as practical viewpoint, is widely accepted in the literature. In the remainder of this paper, we therefore also assume that the part input sequence is the same for each MPS produced. Moreover, we only consider the cycle time minimization objective (thus reflecting our conjecture that the complexity of makespan and cycle time minimization problems are of the same order for the RFS problem).

As a final remark on ordinary flowshop scheduling problems, let us briefly consider problems with few parts or few part types. When there are only few parts (i.e., when the number of parts is bounded by a constant), complete enumeration of all part input sequences yields a polynomial-time solution strategy for the problem. On the other hand, when the number of part types (as opposed to the number of parts) is fixed, then the total number of part input sequences can be exponential in the length of the encoding of an instance and brute

force enumeration becomes inefficient. Therefore, such problems (sometimes referred to as high multiplicity problems) have their own interesting characteristics from the viewpoint of algorithmic complexity; see e.g. Agnetis [1997a] for a more detailed treatment.

By contrast with the above discussion, we shall see that, in robotic cells, nontrivial problems arise even when the number of parts in the minimal part set is small. Actually, even when there is a *unique* part type (a vacuous situation in traditional flowshops), the question remains of specifying an optimal (or feasible) robot move sequence.

Let us now turn to the RFS problem. We organize our discussion into three separate cases, depending on the width of the processing windows.

4.1 Arbitrary processing windows

In this section, we make no special assumption concerning the properties of the processing windows. However, we restrict our discussion to the special case of the RFS problem where all parts are identical and we want to compute a 1-periodic schedule with minimum long run cycle time. As before, we denote the processing windows as $[L_i, U_i]$ for $i = 1, \ldots, n$.

Observe that, for any given 1-unit cycle π, the associated optimal 1-periodic schedule can be computed in polynomial time by solving the linear programming problem

$$\min C_S \quad \text{subject to (7)-(13).} \tag{14}$$

Thus, the special case of (RFS) under consideration essentially boils down to the following:

Definition 6 *Robotic Flowshop Scheduling for Identical Parts* (RFSI):

INPUT : δ_{ij}, ϵ_i, $[L_i, U_i]$ for $i = 0, \ldots, m+1$ and $j = 1, 2, \ldots$;

QUESTION : Find a 1-unit cycle with minimum long run cycle time.

The complexity of this problem has been investigated in a series of papers. Using today's terminology,the decision version of (RFSI) (with arbitrary robot travel times) has been proved to be NP-complete by Livshits et al. [1974]. Crama and Van de Klundert [1997c] established that the decision version of (RFSI) is strongly NP-complete even when the travel times satisfy the triangle equality. Notice that the proof techniques are explicitly limited to the determination of an optimal 1-unit cycle so that, strictly speaking, the complexity of (RFS) remains open even in the case of identical parts (as the optimal sequence may not arise from a 1-unit cycle).

Let us briefly review the literature on solution methods for the (RFSI) problem and closely related variants. The problem, together with the LP formulation of the cycle time subproblem, was introduced independently by Livshits et al. [1974] and Phillips and Unger [1976]. Phillips and Unger formulate (RFSI) as an integer linear program and solve some instances using a commercial software package. Kats [1982], Lei and Wang [1994], Armstrong, Lei and Gu [1994], Chen et al. [1994] and Hanen and Munier [1994] propose branch and bound procedures for (RFSI) and various extensions. Hertz et al. [1996] solve a related problem using tabu search, Varnier et al. [1996] attack the problem by constraint logic programming techniques and Rochat [1995] implements a genetic algorithm.

4.2 Zero-width processing windows

In this subsection, we briefly review the literature on the special case (of RFS) where all parts are identical and $U_i = L_i$ for $i = 1, \ldots, m$. Suprunenko et al. [1962], Aizenshtat [1963a, 1963b] and Tanayev [1964] were apparently the first to consider this problem, more than three decades ago. They developed mathematical models for constructing both 1-unit and multiple-unit schedules. Using the results of Aizenshtat [1963a, 1963b] and Livshits et al. [1974], Kats and Mikhailetsky [1980] determined that the total number of different 1-unit cycles for a given instance is at most $O(m^3)$. They designed a polynomial-time algorithm to compute an optimal 1-unit schedule.

Levner, Kats and Levit [1997] present an algorithm that refines Aizenshtat's 'prohibited interval' rule and solves the problem in $O(m^3 \log m)$ time in the special case where the robot travel times satisfy a version of the triangle inequality. Their approach is to identify a set of 'forbidden intervals' for the cycle time. Based on these intervals, they are able to identify a smallest attainable cycle time and a 1-unit cycle that achieves this cycle time. Their results have been extended in several directions. Kats and Levner [1997a] propose strongly polynomial algorithms for extensions to re-entrant flowshops, while Kats and Levner [1998a] do the same for problems involving more than one robot. Moreover, Kats [1982] and Kats and Levner [1997b] develop an $O(m^5)$ algorithm to simultaneously optimize over the number of robots and the cycle time. Recently, Kats and Levner [1998a] improved their strongly polynomial algorithm for the case where the triangle inequality is not required to hold. They propose an $O(m^3 \log m)$ time algorithm exploiting a neighborhood structure on the m^3 possible robot tours and show how to switch in $O(\log m)$ time from neighbor to neighbor. In their study of the same problem, Song et al. [1993] have noticed that a 2-unit schedule may provide a better cycle time than the best 1-unit schedule and have derived an SPT-type heuristic for solving the problem. The zero-width problem with multiple part types is explored in Agnetis [1997b]. Agnetis shows that the 2-machine problem with h different parts can be solved in $O(h \log h)$ time, using a Gilmore-Gomory based algorithm (cf. Hall et al. [1997]).

4.3 Unbounded processing windows

In this subsection we consider the special case of GRFS where $U_i^j = +\infty$ for all i, j. This special case, in which the machines behave as is customary in classical scheduling problems, has given rise to a very fruitful research area. In the production environments that originally motivated this line of research, the machines are placed in a line or a circle and, therefore, the robot travel times are usually assumed to satisfy the triangle equality. Throughout this subsection, we shall again assume it to hold.

In a seminal paper, Sethi et al. [1992] introduced the problem and studied special cases involving only two or three machines. They showed that, in two-machine shops, the problem of finding an optimal part input sequence corresponding to a fixed 1-unit cycle can be solved in $O(h \log h)$ time, where h is the number of parts in an MPS. Their algorithm is an extension of the Gilmore-Gomory algorithm for the classical two-machine bufferless flowshop (Gilmore and Gomory [1964]). They also explain how the problem can be solved by enumeration when there is only one part type and three machines. Moreover, they derive that, in the latter case, two of the six possible 1-unit cycles are always dominated by the remaining four ones. These results have been extended in two directions.

As shown in Crama and Van de Klundert [1997a], the results for identical parts can be generalized. Indeed, in a robotic cell with m machines, there exists a set of 1-unit cycles of cardinality 2^{m-1} that necessarily contains an optimal solution. This is the set of so-called 'pyramidal permutations' of the activities. Pyramidal permutations have previously been investigated in the context of polynomially solvable cases of the traveling salesman problem, in the same family that also contains the two-machine flowshop problem solvable by the previously mentioned Gilmore Gomory algorithm [Gilmore et al. 1985]. Crama and Van de Klundert [1997a] give algorithms which compute an optimal 1-unit cycle in $O(m^3)$ or $O(m^2 \log(dm))$ time, where $d = \max_{i,j} \delta_{ij}$. On the negative side, Brauner, Finke and Kubiak [1997] established that the decision version of the problem becomes strongly NP-complete if the travel times do not satisfy the triangle equality.

Hall et al. [1995,1997], Kamoun et al. [1993], Sriskandarajah et al. [1998] investigated problems with multiple part types. Hall et al. [1997] show that in a two-machine flowshop, it is possible to simultaneously optimize the part input sequence and the h-unit cycle in polynomial time. Their algorithm uses again the Gilmore-Gomory algorithm and identifies an optimal way to switch between the two possible 1-unit cycles. This appears to be the first algorithm that simultaneously optimizes over part input and robot move sequences. These authors also show that the problem is NP-hard for robotic flowshops with three or more machines. More precisely, for each possible 1-unit cycle in an m-machine flowshop, they either give an algorithm which computes the optimal part input sequence in time polynomial in m, or they show that it is NP-hard to find the optimal part input sequence.

Finally, Chen et al. [1994] describe and test a branch and bound algorithm based on the Gilmore-Gomory algorithm for the multipart problem in three-machine flowshops.

5 Cycle time computation

5.1 Introduction

This section studies the problem of computing the long run cycle time when both the part input sequence and robot move sequence are fixed. If the robot executes a k-unit cycle and h is the number of parts in the MPS, we shall see that there exists a $(k \times h)$-periodic schedule that achieves the minimum long run cycle time. Thus, the problem of computing the long run cycle time boils down to solving a linear programming problem similar to (14) and we conclude that the cycle time problem is polynomially solvable.

In many cases, however, the cycle time can be computed much more efficiently than by solving an LP model. To understand this, observe first that, when the part input sequence and the robot move sequence are specified, all precedence relations become fixed. In many classical scheduling problems, the makespan can then be derived by performing a PERT/CPM analysis or, more specifically, by calculating the length of a longest path in a directed graph. Such graphs are usually acyclic since their arcs represent precedence relations.

In the same spirit, the cycle time of a cyclic scheduling problem (not necessarily a robotic flowshop problem) can be determined by analyzing a cyclic graph (i.e., a graph containing at least one cycle) in which the cycles are created by precedence relations between tasks to be performed in successive repetitions of the schedule (see for instance McCormick et al. [1989]). In such cyclic graph models, the cycle time turns out to be equal to the weight of a maximum weight cycle (instead of the length of a longest path). Since such cyclic

PERT models are applicable to a variety of cyclic scheduling problems, they have been studied by several authors, in other environments than robotic cells. We mention cyclic staff scheduling problem (Bartholdi et al. [1980]), cyclic job shop scheduling (Roundy[1992], Lee and Posner [1997]), scheduling of MPSs through a flexible flowshop (Matsuo et al. [1991]), scheduling problems arising in the optimization of compilers for parallel processors (Hanen and Munier [1995]), and of course robotic flowshop scheduling. Timkovsky [1986] studies cycle time minimization and schedule feasibility problems for re-entrant (no-robot) flowshops and derives a fast algorithm for minimizing the cycle time in a problem with unbounded processing windows. His results are closely related to the cycle time minimization algorithm in Pinedo [1995, page 265] and the results mentioned in section 5.2.

As is the case for the closely related min cost circulation flow problem, there are many solution methods available for the cyclic PERT problem and oftentimes different authors have taken different routes. As already mentioned above, several authors use minimum mean cycle based approaches (Karp [1978], Cohen et al. [1984] Matsuo [1991] and Van de Klundert [1996]). Karp and Orlin [1981], Hartmann and Orlin [1993], Kats and Levner [1996], Lee and Posner [1997] use a parametric shortest path approach. Other solution techniques that have been applied are Bellman-Ford node labeling approaches (Chen et al. [1994], Ioachim and Soumis [1995], Kats and Levner [1996]), a network simplex approach (Roundy [1991], binary search (Gondran and Minoux [1984], Lei [1991]) or special purpose algorithms (Ng and Lueng [1997]).

The layout of this section will be somewhat different from the layout of the previous one. First of all, we do not consider the cycle time minimization problem in the case of zero-width processing windows, for the following reason. In applications involving zero-width windows, it is customary to simultaneously impose loaded-robot restrictions (see Section 2.2), so that all inequalities (7)-(11) must be satisfied at equality. The cycle time can then be easily determined in quadratic time by solving a system of equations (see Levner, Kats and Levit [1997]). Furthermore, we treat the special case of unbounded processing windows before the more general case, since this facilitates the exposition. Finally, for the sake of simplicity and without loss of generality, we assume in the remainder of this section that $\epsilon_i = 0$ for $i = 0, \ldots, m+1$.

5.2 Unbounded processing windows

Assume that all the processing windows have an infinite upper-bound and let us first consider the simple case where all parts are identical and the robot repeatedly performs a 1-unit cycle π. Then, the system of inequalities (7)-(13) describes the set of feasible 1-periodic schedules. In fact, since $U_i = +\infty$ for $i = 1, \ldots, m$, the constraints (8) and (10) can even be eliminated from this formulation.

Let us now select a tentative value C_S for the cycle time and let us build a directed graph $G = (V, A)$ as follows. The graph G has one vertex for each load operation and one vertex for each unload operation. Moreover, each of the inequalities (7), (9) and (11)-(13) gives rise to an arc whose length corresponds to the right-hand side of the inequality (observe that the arc lengths depend on the value selected for C_S).

Consider for example the case where the 1-unit cycle is $\pi = (A_0, A_2, A_1, A_3)$. Then the corresponding robot travel time constraints (11)-(13) are represented in Figure 3.

Notice that, except for the arc arising from constraint (13), this graph is acyclic. Next,

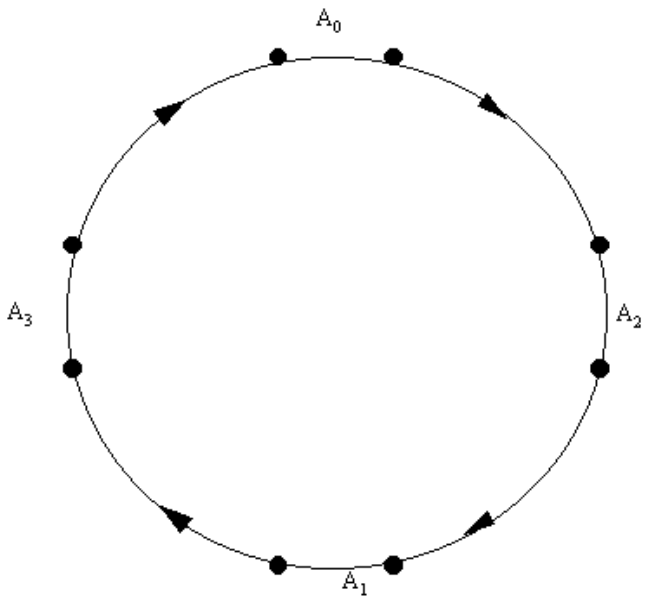

Figure 3: cycle time computation graph, robot travel time

we add the arcs representing the constraints (7) and (9), where the latter may again induce cycles (see Figure 4).

We will refer to the arcs associated to constraints (9) and (13) as *backward* arcs. Let us now consider any two vertices of G, say v and w. As the reader may check, the length of any path from v to w that does not contain any backward arcs is a lowerbound on the time that must elapse between the load/unload operations represented by v and w.

Consider next a simple cycle in G and let b be the number of backward arcs in this cycle. Further, assume that the value of C_S is such that the cycle has positive length. Then, as observed by many authors, C_S cannot be a feasible cycle time. Indeed, the length of the cycle plus $b \times C_S$ is a lowerbound on the amount of time that must elapse between the first and $(b+1)$-th execution of the operation corresponding to any vertex on the cycle. Thus, there follows that C_S cannot be feasible if the cycle has positive length. From the form of inequalities (9) and (13), it is clear that this infeasibility can only be remedied by increasing the cycle time C_S.

Reasoning along these lines, the following theorem can be formally established (see for instance Van de Klundert [1996]):

Proposition 3 The minimum long run cycle time of a 1-unit cycle π is equal to the minimum value of C_S for which the graph G contains no cycles of positive length. Moreover, there always exists a 1-periodic schedule whose cycle time achieves the minimum long run cycle time of π.

As a consequence of this result, the computation of the optimal schedule essentially reduces to the computation of the minimum cycle time. Cohen et al. [1985] and Carlier and Chrétienne [1988] were apparently the first who noticed that Karp's algorithm [1978] for

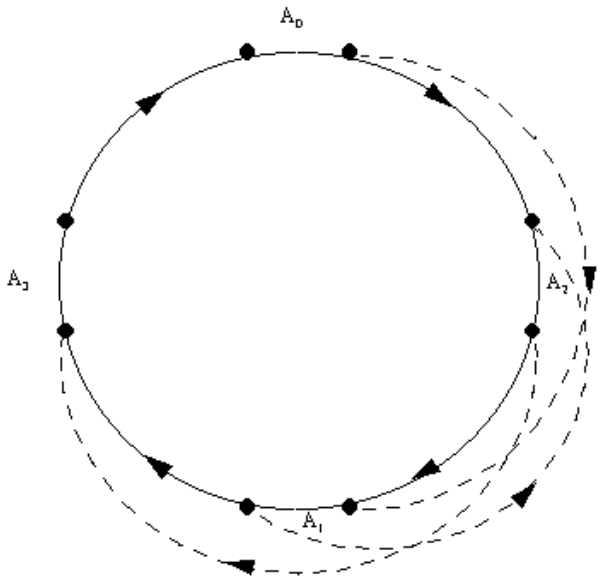

Figure 4: cycle time computation graph, unbounded time windows

finding a minimum mean-length cycle in a digraph can be used as a 'blackbox' for solving the aforementioned problem. This approach leads to an $O(m^3)$ algorithm for the computation of an optimal schedule. Similar observations were also exploited by Matsuo et al. [1991] and Van de Klundert [1996].

Another approach was developed by Ioachim and Soumis [1995]. It is based on iterative use of the Bellman-Ford algorithm for computing the critical (longest) path in a network and leads again to $O(m^3)$ time complexity.

Kats and Levner [1996] and Lee and Posner [1997] have proposed yet another tool for the same problem: they rely on the Karp-Orlin algorithm [1981] for finding the parametric shortest path in a graph. This allows them to compute an optimal schedule in $O(m^2 \log m)$ time.

Roundy's approach [1992] to cyclic job shop scheduling yields an $O(m^2 \log m \log B)$ algorithm for our problem, where B is a (polynomial) upperbound on the value of the optimal solution. This approach is based on the use of a network simplex algorithm. Finally, we observe that a recent algorithm proposed by Hartmann and Orlin [1993] (in the context of crew scheduling) translates into a current best $O(m^2)$ time algorithm for the computation of the minimum cycle time in the case of unbounded windows.

So far in this section, we have concentrated on the situation where all parts are identical and the robot executes a 1-unit cycle. Let us now briefly turn to the general formulation of the problem.

First of all, the graph model presented above can be extended without difficulty to the case where the robot repeatedly executes a k-part cycle and the MPS consists of h parts. In this case, the graph contains $k \times h$ vertices for each load/un load operation. (In fact, instead of $k \times h$, the smallest common multiple of k and h would suffice.) Moreover, the previous

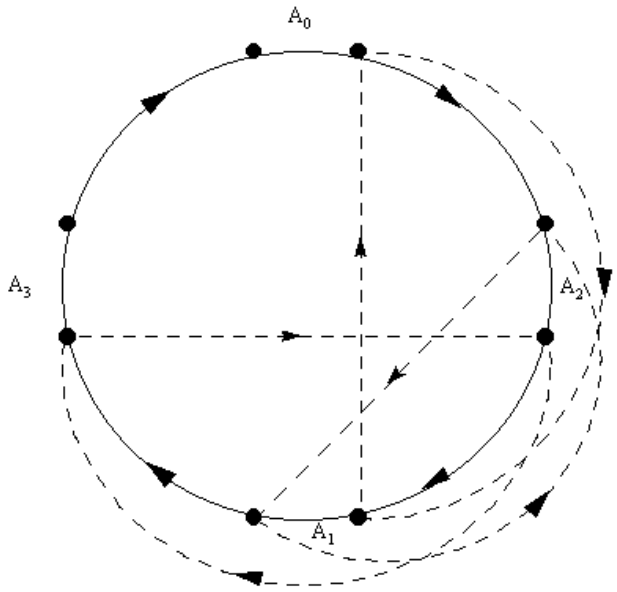

Figure 5: Cycle time computation graph, general case

analysis can be extended to show that there exists a $k \times h$-periodic schedule that achieves the minimum long run cycle time (Van de Klundert [1996]).

Matsuo et al. [1991] showed that the dual of the LP formulation is a maximum cost circulation problem with the additional constraint that the sum of the flows over all backward arcs equals one. Matsuo et al. [1991] and Van de Klundert [1996] described an $O(hm^3)$ algorithm for the special, but practically important, case where the MPS consists of h parts *and* the robot performs a h-unit cycle. Ioachim and Soumis [1995] proposed an $O(h^3m^3)$ for the same problem.

5.3 Arbitrary processing windows

The case of arbitrary processing windows differs from the case treated in the previous subsection in the sense that constraints (8) and (10) are now also in force. Therefore, the arcs representing these constraints must also be added to the graph G introduced above. For the example treated in Section 5.2, we obtain the graph of Figure 5.

Notice that this graph does not only contain backward arcs, i.e. arcs whose length includes a term '$-C_S$', but also arcs that might be refered to as forward arcs, i.e. arcs whose length includes a term '$+C_S$. When all parts are identical and the robot performs a 1-unit cycle, it is again possible to prove that there exists a 1-periodic schedule attaining the minimum long run cycle time. Moreover, it can be shown that in the general case where the robot repeatedly excutes a k-unit cycle and the MPS consists of h parts, there exists an optimal $(k \times h)$ periodic schedule.

Lei [1993] described a linear programming formulation of the problem, similar to the one given above. In the case of integer data, she also presented a (weakly) polynomial algorithm

based on binary search. The time complexity of this algorithm is $O(m^2 \log m \log B)$, where B is an a priori bound on the range of possible cycle times such that $\log B$ depends linearly on the size of the input. Chen, Chu and Proth [1994] derived an $O(m^6)$ algorithm based on a straighforward appliction of the Bellman-Ford longest path algorithm. Kats and Levner [1996] proposed an $O(m^3)$ algorithm based on a parametric modification of the Bellman-Ford algorithm (for finding the critical path) which can treat any *real* input data. Both these results are only valid for the case of identical parts and 1-unit cycles, but can be straightforwardly generalized to deal with k-unit cycles, h-part MPSs and $(k \times h)$-periodic schedules (although we could not find this observation in the literature).

6 Other topics and further research

This survey discusses cycle time minimization problems as they occur in robotic cells of different types. We have attempted to describe the state-of-the-art concerning various models and to give a classification of the problems. As yet, we can claim that the most basic and important problems are relatively well understood, at least from a complexity viewpoint. However, several interesting problems still remain open. In this section, we formulate what we think to be the most tempting areas for further research.

The results in Section 4 leave open the complexity of robotic cell scheduling problems with a constant number of part types. In such a case, the description of the input only requires an amount of space that is logarithmic in the number of parts. Hence, any polynomial algorithm for such problems should have running time that is only logarithmic in the number of parts. At this moment, there are no specialized algorithms available for such problems, even if we allow their running time to be polynomial in the number of parts. In practice, it may often be the case that the number of parts in the minimal part set is small. Therefore, algorithms whose complexity is exponential in the number of part types and polynomial in (the logarithm of) the number of parts may still be of practical interest.

The most severe restrictions we placed on (certain) models consisted in considering only 1-unit and 1-periodic schedules. As we have seen, these restrictions often allowed for efficient solution strategies, but may have excluded better solutions beforehand. Moreover, even though the description of sequences should preferably be short, there may be short sequences that are not 1-unit sequences and that are well worth considering in practical applications (see e.g. Song et al. [1993], Lei and Wang [1994]). Kats, Levner and Meyzin [1997] study a mathematical model for obtaining optimal k-unit cyclic schedules and obtain provably optimal solutions for some test and benchmark problems.

Let us return to results that justify restrictions of the aforementioned type, in the case of unbounded processing windows. The strongest result in this area is probably the algorithm due to Hall et al. [1997], which computes simultaneously the optimal h-unit feasible robot move sequence and the optimal part input sequence for two-machine flowshops. The results in the previous section show that h-periodic schedules are optimal over all schedules for h-unit cycles in the same environment. Sethi et al. [1992] conjectured that, for three-machine cells and identical parts, 1-unit robot move sequences are always optimal in case the robot travel times satisfy the triangle equality. Hall et al. [1995] proved that the conjecture holds in several special cases. More recently, Crama and Van de Klundert [1997b] established the validity of the conjecture. An alternative, more compact proof was obtained by Brauner and

Finke [1997a]. Crama and Van de Klundert [1997b] in turn conjectured that 1-unit robot move sequences are optimal when all parts are identical, whatever the number of machines is. The validity of this conjecture would imply that the algorithm proposed by Crama and Van de Klundert [1997a] solves the robotic flowshop problem with identical parts and unbounded time windows (in which the triangle equality holds) to optimality, even if the 1-unit cycle restriction is relaxed. However, this conjecture was recently disproved by Brauner and Finke [1997b] for the 4-machine case.

Several other versions of the conjecture had already been previously disproved. For instance, Hall et al. [1997] showed that, in case of multiple part types, the set of 1-unit robot move cycles does not necessarily contain an optimal solution. Also, Hertz [1995] observed that the conjecture may fail when the triangle equality is violated.

In the case of finite upper bounds on the processing windows, we also restricted our discussion to 1-periodic schedules. It is known, however, that both in the general case (Lei [1995]) and in the zero-width case (Levner, Kats and Sriskandarajah [1996]), the set of 1-unit cycles does not necessarily contain an optimal solution, even for identical parts.

For 3-machine cells with no-wait restrictions on the robot, zero-length processing windows and identical parts, Agnetis [1997b] showed that the optimal cycle is either a 1-unit or a 2-unit cycle. Moreover, he conjectures that, for m-machine cells, the minimum cycle time can be achieved by a k-unit cycle with $k < m$.

A number of open problems arise from this overview.

1. When 1-unit cycles are not optimal, can one derive a tight upper bound c such that there always exists an optimal k-unit cycle with $k \leq c$. Even for identical parts, when the data are arbitrary real numbers, it is not known whether there always exists a finite value of k such that some k-unit cycle achieves the minimum cycle time.

2. When 1-unit cycles are not optimal, what is the complexity of finding the optimal k-unit cycle? Currently, the complexity of finding the optimal 2-unit cycle appears to be open, even when all data are integer.

3. When 1-unit cycles are not optimal, derive a tight bound on their relative performance.

4. Is the cycle time computation problem for general processing windows really harder than for unbounded windows, or do both cases have the same time complexity? In particular, is it possible to solve the general problem in $O(m^2)$ time? Is it possible to obtain more efficient algorihms for cycle time computation problems, e.g. by transforming the cycle time computation problem to a (maximum cost) network flow problem?

7 Appendix

Proof of Proposition 1 The condition is clearly necessary, since between every two consecutive occurences of A_i, machine M_i must be loaded and machine M_{i+1} must be unloaded exactly once.

In order to prove sufficiency, let us start by defining the initial state of the flowshop. Without loss of generality, assume that, when the robot starts executing π, machine M_i holds a part if and only if the first occurrence of A_i precedes the first occurrence of A_{i-1} in π, for $i = 1, \ldots, m$ (such a machine must be unloaded before it is loaded).

A moment of reflection then shows that, in the course of executing π, the robot only unloads machines that actually contain a part and only loads machines that are empty (for the first execution of each activity, this holds true by construction of the initial state; for subsequent activities, it holds true by the property of π). □

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). The fourth author gratefully acknowledges partial support by the Ministry of Science, Israel, within the cooperations program with the Science and Technology Agency of Japan (grant 8951197) and the French-Israeli cooperation program Arc-en-Ciel/Keshet (grant 1997-25).

8 References

Agnetis, A., 1997a, No-wait flowshop scheduling with large lot size, *Annals of Operations Research* 70, 415-438.

Agnetis, A., 1997b, Scheduling no-wait robotic cells with two and three machines, Technical Report 21-97, Universita degli studi di Roma, La Sapienza.

Aizenshtat, V.S., 1963a, Combining primitive cyclic processes, *Doklady Academy Nauk BSSR* 7(3), 148-151 (in Russian).

Aizenshtat, V.S., 1963b, Multioperator cyclic processes, *Doklady Academy Nauk BSSR* 7(4), 224-227 (in Russian).

Armstrong, R., Lei, L., Gu, S., 1994, A bounding scheme for deriving the minimal cycle time of a single transporter N-stage process with time windows, *European Journal of Operational Research* 78, 130-140.

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

Bartholdi, J.J., Orlin, J.B., Ratliff, H.D., 1980, Cyclic scheduling via integer programs with circular ones, *Operations Research* 28, 1073-1085.

Blazewicz, J., Finke, G., 1994, Scheduling with resource management in manufacturing systems, *European Journal of Operational Research* 76, 1-14.

Brauner, N., Finke, G., 1997a, On the conjecture in robotic cells: new simplified proof for the three machine case, Rapport de Recherche RR 971-I, Laboratoire Leibniz, Institut Imag, Grenoble.

Brauner, N., Finke, G., 1997b, Final results on the 1-cycle conjecture in robotic cells, Manuscript, Laboratoire Leibniz, Institut Imag, Grenoble.

Brauner, N., Finke, G., Kubiak, W., 1997, A proof of the Lei and Wang claim, Technical Note, Laboratoire Leibniz, Institut Imag, Grenoble.

Carlier, J., Chrétienne, Ph., 1988, *Problèmes d'Ordonnancement: Modélisation, Complexité, Algorithmes,* Etudes et Recherche en Informatique, Ed. Masson, Paris.

Chen, H., Chu, C., Proth, J.-M., 1994, Cyclic scheduling of a hoist with time window constraints, Rapport de recherche Nr 2307, INRIA.

Chu, C., Proth, J.-M., 1996, Single machine scheduling with chain structures precedence constraints and separation time windows, *IEEE Transactions on Robotics and Automation* 12(6), 835-844.

Cohen, G., Dubois, D., Quadrat, J.P., Viot, M., 1985, A linear-system-theoretic review of discrete-event processes and its use for performance evaluation in manufacturing, *IEEE Transactions on Automatic Control* AC30(3), 210-220.

Crama, Y., 1997, Combinatorial optimization models for production scheduling in automated manufacturing systems, *European Journal of Operational Research* 99, 136-153.

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

Crama, Y., Van de Klundert J., 1997b, Cyclic scheduling in 3-machine robotic flow shops, Research memorandum RM97018, Maastricht University, Maastricht. In revision for *Journal of Scheduling.*

Crama, Y., Van de Klundert J., 1997c, Robotic flowshop scheduling is strongly NP-complete, in: *Ten Years LNMB,* W.K. Klein Haneveld et al. (eds.), CWI Tract, Amsterdam, The Netherlands, 277-286.

Finke, G., Gueguen, C., Brauner, N., 1996, Robotic cells with buffer space, in: *ECCO IX Conference Proceedings,* Rory O'Conner and Patricia Magee (eds.), Dublin City University.

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

Gilmore, P.C., Lawler, E.L., Shmoys D.B., 1985, Well solved special cases, in: *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization,* E.L. Lawler et al. (eds.), Wiley-Interscience Series in Discrete Mathematics, John Wiley and Sons, Chichester New York Brisbane Toronto Singapore.

Glass, C.A., Shafransky, Y.M., Strusevich, V.A., 1995, Scheduling for parallel dedicated machines with a single server, CASSM R&D Paper 8, University of Greenwich, London.

Gondran, M., Minoux, M., 1984, *Graphs and Algorithms,* Wiley Interscience, Chichester New

York Brisbane Toronto Singapore.

Hall, N.G., Kamoun, H., Sriskandarajah, C., 1995, Scheduling in robotic cells: complexity and steady state analysis, Working Paper, College of Business, The Ohio State University.

Hall, N.G., Kamoun, H., Sriskandarajah, C., 1997, Scheduling in robotic cells: classification, two and three machine cells, *Operations Research* 45(3), 421-439.

Hall, N.G., Potts, C.N., Sriskandarajah, C., 1995, Parallel machine scheduling with a common server, Working Paper, The Ohio State University.

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

Hanen, C., Munier, A., 1994, Periodic scheduling of several hoists, in: *Proc. Fourth International Conf. Project Management and Scheduling,* 108-110.

Hanen, C., Munier, A., 1995, Cyclic scheduling on parallel processors: an overview, in: *Scheduling Theory and its Applications,* Ph. Chrétienne et al. (eds.), John Wiley & Sons.

Hartmann, M., Orlin, J.B., 1993, Finding minimum cost to time ratio cycles with small integer transit times, *Networks* 23, 567-574.

Hertz, A., 1995, private communication.

Hertz, A., Mottet, Y., Rochat, Y., 1996, On a scheduling problem in a robotized analytical system, *Discrete Applied Mathematics* 65, 285-316.

Hitz, K.L., 1980, Scheduling of flow shops II, Report No. LIDS-R-879, Laboratory for information and decision systems, M.I.T., Cambridge (Mass.).

Ioachim, I., Soumis, F., 1995, Schedule eficiency in a robotic production cell, *International Journal of Flexible Manufacturing Systems* 7, 5-26.

Jeng, W.D., Lin, J.T. Wen, U.P., 1993, Algorithms for sequencing robot activities in a robot-centered parallel-processor workcell, *Computers and Operations Research* 20(2), 185-197.

Johnson, S.M., 1954, Optimal two and three-stage production schedules with set up times included, *Naval Research Logistics Quarterly* 1, 61-68.

Kamoun, H., Hall., N.G., Sriskandarajah, C., 1993, Scheduling in robotic cells: heuristics and cell design, Working Paper, University of Toronto. To appear in *Operations Research.*

Kamoun, H., Sriskandarajah, C., 1993, The complexity of scheduling jobs in repetitive manufacturing systems, *European Journal of Operational Research* 70, 350-364.

Karp, R.M., 1978, A characterization of the minimum cycle mean in a digraph, *Discrete Mathematics* 23, 309-311.

Karp, R.M., Orlin, J.B., 1981, Parametric shortest path algorithms with an application to cyclic staffing, *Discrete Applied Mathematics* 3, 37-45.

Karzanov, A.V., E.M. Livshits, 1978, Minimal quality of operators for serving a homogeneous linear technological process, *Automation and Remote Control* 39(3), Part 2, 445-450.

Kats, V.B., 1982, An exact optimal cyclic scheduling algorithm for multioperator service of a production line, *Automation and Remote Control* 42(4), Part 2, 538-543.

Kats, V., Levner, E., 1996, Polynomial algorithms for scheduling of robots, in: *Intelligent Scheduling of Robots and FMS,* E. Levner (ed.), CTEH Press, Holon, Israel, 77-100.

Kats, V., Levner, E., 1997a, A strongly polynomial algorithm for no-wait cyclic robotic flow-shop scheduling, *Operations Research Letters*, 21, 1997, 171-179.

Kats, V., Levner, E., 1997b, Minimizing the number of robots to meet a given schedule, *Annals of Operations Research* 69, 209-226.

Kats, V., Levner E., 1998a, Polynomial algorithms for cyclic scheduling of tasks on parallel processors, Proceedings of the 16th IASTED International conference on applied informatics, February 23-25 1998, Garmisch, Germany, IASTED/ACTA Press, Anaheim, 302-304.

Kats, V. Levner, E., 1998b, Cyclic scheduling of operations for a part type in an FMS handled by a single robot: a parametric critical path approach, *The International Journal of Flexible Manufacturing Systems* 10, 129-138.

Kats, V., Levner, E., Meyzin, L., 1997, Multiple part cyclic hoist scheduling using a sieve method, Technical Report, Center for Technological Education, Holon, Israel.

Kats, V., Mikhailetsky, Z.N., 1980, Exact solution of a cyclic scheduling problem, *Automation and Remote Control* 4, 187-190 (in Russian).

King, R.E., Hodgson, T.J., Chafee, F.W., 1993, Robot task scheduling in a flexible manufacturing cell, *IIE Transactions* 25(2), 80-87.

Kise, H., 1991, On an automated two-machine flowshop scheduling problem with infinite buffer, *Journal of the Operations Research Society of Japan* 34(3), 354-361.

Kise, H., Shioyama, T., Ibaraki, T., 1991, Automated two machine flowshop scheduling: a solvable case, *IIE Transactions* 23(1), 10-16.

Klundert, J.J. van de, 1996, Scheduling problems in automated manufacturing, Dissertation 96-35, Faculty of Economics and Business Administration, Maastricht University.

Kogan, K., Levner, E., 1997, A polynomial algoritm for scheduling small-scale manufacturing cells served by multiple robots, *Computers and Operations Research*, 25, (1), 53-62.

Lee, T.E., Posner, M.E., 1997, Performance measures and schedules in periodic job shops, *Operations Research* 45(1), 72-91.

Lei, L., 1993, Determining the optimal starting times in a cyclic schedule with a given route, *Computers and Operations Research* 20(8), 807-816.

Lei, L., 1995, private communication.

Lei, L., Armstrong, R., Gu, S., 1993, Minimizing the fleet size with dependent time-window and single-track constraints, *Operations Research Letters* 14, 91-98.

Lei, L., Wang, T.J., 1989, A proof: The cyclic hoist scheduling problem is NP-complete, Working Paper 89-16, Graduate School of Management, Rutgers University.

Lei, L., Wang, T.J., 1991, The mimimum common cycle algorithm for cyclic scheduling of two hoists with time window constraints, *Management Science* 37(12), 1629-1639.

Lei, L., Wang, T.J., 1994, Determining optimal cyclic hoist schedules in a single-hoist electroplating line, *IIE Transactions* 26(2), 25-33.

Levner, E., 1969, Optimal planning of parts machining on a number of machines, *Automatic Remote Control* 12, 1972-1978.

Levner, E., Kats, V., Levit, V.E., 1997, An improved algorithm for cyclic flowshop scheduling in a robotic cell, *European Journal of Operational Research* 97, 500-508.

Levner, E., Kats, V., Sriskandarajah, C., 1996, A geometric algorithm for scheduling of robots, in: *Intelligent Scheduling of Robots a and FMS,* E. Levner (ed.), CTEH Press, Holon, Israel, pp. 101-112.

Levner, E., Kogan, K., Levin, Y., 1995, Scheduling a two-machine robotic cell: a solvable case, *Annals of Operations Research* 57, 217-232.

Levner, E., Kogan, K., Maimon, O., 1995, Flowshop scheduling of robotic cells with job dependent transportation and set-up effects, *Journal of the Operational Research Society* 46, 1447-1455.

Levner, E., Vlach, M., 1997, Single machine scheduling with fuzzy precedence constraints, IS-RR-97-0031F, School of Information Science, Japan Institute of Science and Technology, Hokuriku.

Lieberman, R.W., Turksen, I.B., 1981, Crane scheduling problems, *AIIE Transactions* 13(4),

304-311.

Livshits, E.M., Mikhailetsky, Z.N., Chervyakov, 1974, A scheduling problem in an automated flow line with an automated operator, *Computational Mathematics and Computerized Systems* 5, 151-155 (in Russian).

Matsuo, H., Shang , J.S., Sullivan, R.S., 1991, A crane scheduling problem in a computer integrated manufacturing environment, *Management Science* 37(5), 587-606.

McCormick, S.T., Pinedo, M.L., Shenker, S., Wolf, B., 1989, Sequencing in an assembly line with blocking to minimize cycle time, *Operations Research* 37(6), 925-935.

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

Ng, W.C., Leung, J., 1997, Determining the optimal move times for a given cyclic schedule of a material handling hoist, *Computers and Industrial Engingeering* 32(3), 595-606.

Orlin, J.B., 1982, Minimizing the number of vehicles to meet a fixed periodic schedule: An application of periodic posets, *Operations Research* 30(4), 760-776.

Papadimitrou, C.H., Kanellakis, P.C., 1980, Flowshop scheduling with limited temporary storage, *Journal of the ACM* 27(3), 533-549.

Park, Y.B., 1994, Optimizing robot's service movement in a robot-centered FMC, *Computers and Industrial Engineering* 27(1-4), 47-50.

Phillips, L.W., Unger P.S., 1976, Mathematical programming solution of a hoist scheduling program, *AIIE Transactions* 8(2), 219-225.

Pinedo, M., 1995, *Scheduling: Theory, Algorithms and Systems,* Prentice-Hall, Englewood Cliffs (NJ).

Rochat, Y., 1995, A genetic approach for solving a scheduling problem in a robotized analytical system, in: *Intelligent Scheduling of Robots and FMS,* E. Levner (ed.), Center for Technological Education Holon, Israel, pp. 191-209.

Rock, H., 1984, The three-machine no-wait flowshop scheduling problem is NP-Complete, *Journal of the ACM* 33, 336-345.

Roundy, R., 1992, Cyclic schedules for job-shops with identical jobs, *Mathematics of Operations Research* 17, 842-865.

Serafini, P., W. Ukowich, 1989, A mathematical model for periodic scheduling problems, *SIAM Journal on Discrete Mathematics* 2, 550-581.

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

Shapiro G.W., Nuttle, H.L.W., 1988, Hoist scheduling for a PCB electroplating facility, *IIE Transactions* 20, 157-167.

Sriskandarajah, C., Hall., N.G., Kamoun, H., Wan, H., 1998, Scheduling large robotic cells without buffers, *Annals of Operations Research* 76, 287-321.

Song, W., Zabinsky, Z.B., Storch., R.L., 1993, An algorithm for scheduling a chemical processing tank line, *Production Planning and Control* 4(4), 323-332.

Suprunenko D.A., Aizenshtat V.S., Metel'skii A.S., 1962, A multistage technological process, *Doklady Academy Nauk BSSR* 6(9), 541-544 (in Russian).

Tanayev, V.S., 1964, A scheduling problem for a flowshop line with a single operator, *Eng.-Physics Journal* 7(3), 111-114 (in Russian).

Timkovsky, V., 1986, An approximate solution of a scheduling problem in a cyclic system, *Economics and Mathematical Methods* 8, 171-174 (in Russian).

Varnier, C., Bachelu, A., Baptiste, P., 1996, A hoist scheduling problem application in chemical production line, in: *Intelligent Scheduling of Manufacturing Systems,* E. Levner (ed.), Institute of Technology, Arts and Science, Holon, Israel, pp. 267-282.