Robotic flowshop scheduling is strongly NP-complete

Robotic flowshop scheduling is strongly NP-complete

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 flowshop model in which a single robot is responsible for the transportation of parts between machines and the amount of time that a part spends on a machine must be comprised in some predefined interval. The objective is to find a feasible schedule with minimal cycle time. Many researchers have proposed nonpolynomial solution methods for a variety of closely related robotic flowshop scheduling problems. This paper provides a proof that a basic version of this problem is strongly NP-Complete.

1 Introduction

One of the offsprings of the Just In Time production philosophy was the introduction of so called One-Worker Multiple-Machine lines, in which several machines are encircling a single operator. Typically, in the highly repetitive manufacturing environments of Just In Time implementors, all products, or parts, enter the line at an input or input/output station, and require processing on every machine in a prescribed order that is identical for all parts. Finally, the parts are delivered at the output (or input/output) station. The Just In Time emphasis on eliminating inventory demanded that in such a production cell, inventory may only be kept at the input/output station(s). As the automation of production advanced, the operator in the center of the cell, who performed materials handling activities, machine setups and quality inspection, has often been replaced by a single robot. The resulting production cell, is often referred as *robotic flowshop* (see Figure 1). Scheduling problems in such cells have become known as robotic flowshop scheduling

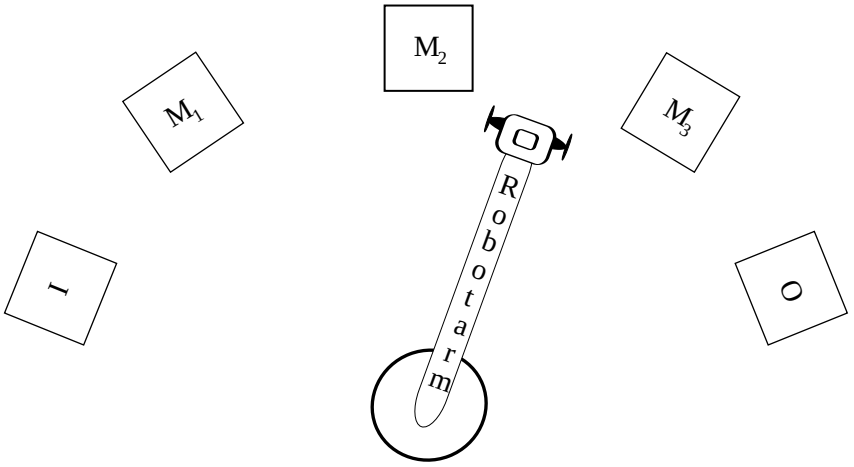

Figure 1: A 3-machine robotic flowshop

problems, robotic cell scheduling problems, and crane or hoist scheduling problems.

Such a robotic flowshop can be viewed as a small fully automated, and therefore unmanned, manufacturing system, moreover such small production systems are the building blocks of larger automated manfacturing systems. As such, an understanding of the problems arising when planning, scheduling and controling the activities in a robotic flowshop are a prerequisite for efficiently operating many large and expensive automated manufacturing systems. The volumes required to earn back the high purchasing costs of such automated manufacturing systems can often only be achieved when producing for a world wide market, such as the automobile industry or consumer electronics. Typically, these markets are characterized by heavy cost based competition, which poses high demands on production efficiency.

The automation of manufacturing in general has led to a variety of problems in which not only the scheduling of different subsystems (e.g. materials handling systems, tool handling systems, flexible machines), but also the interactions of these subsystems with other subsystems are crucial for the overall production efficiency. Scheduling problems in which the coordination between several automated systems is important have therefore received considerable attention lately in the operations research and, more specifically, scheduling literature. The recently widely investigated robotic flowshop scheduling problems (see Asfahl [1985], Crama and Van de Klundert [1994], Hall et al [1994], Sethi et al [1992], Levner et al [1996], Lei and Wang [1994] and the references therein) form a family of such scheduling problems, in which the interaction between the materials handling system and the machines processing the parts determines the production efficiency.

In the next section we describe a basic scheduling problem that has been widely investigated and contains many related problems as a special case. Further we briefly review in section 2, the literature on this problem. In Section 3 we show that the problem is strongly NP-Complete, which provides justification for the nonpolynomial and/or approximate solution methods proposed by several authors.

2 Robotic flowshop Scheduling Models

To formalize and make precise our description of robotic flowshops, we introduce first some notation. We consider robotic flowshops in which there is an input station I or M_0 and a separate output station O or M_{m+1}. Further, there are m machines $M_1, \ldots, M_m$. All machines $M_1, \ldots, M_m$ can contain only one part at a time, and hence when a machine has finished processing a part, the robot must unload it before the machines starts processing the next one. The robot can only carry one part at a time. Every part becomes available at M_0, and requires processing on every machine $M_i, i = 1, \ldots, m$, in increasing order of the the indices of the machines. Finally, each part must be delivered at the output device M_{m+1}. There are no buffers in the flowshop, which yields that each part that is between the input and output stations is either at some machine or being carried by the robot. The robot performs four types of operations. As already mentioned, it unloads parts from machines. When a part is unloaded, it is carried to the next machine, and subsequently loaded there. Finally, the robot will be repositioned to be ready to unload another machine et cetera.

Since we are interested in establishing a NP-Completeness proof, we consider a problem that is general enough to contain several widely investigated problems as a special case, but at the same time contains the properties that are widely considered to be characteristic for robotic flowshop scheduling problems. Further we are of course interested in identifying the 'easiest' problem variation that is already strongly NP-Complete.

As a first step, we therefore restrict the analysis to the case where all parts are identical, i.e. have the same processing requirements. Notice that this eliminates entirely the part sequencing decisions that constitute a classical flowshop scheduling problem. The resulting complexity is in the robot sequencing, and its interaction with the machines.

A second restriction concerns the robot and the sequence of operations that it executes. From the problem description given above, it should be clear that the robot executes subsequences of operations unload machine M_i, carry the part to machine M_{i+1}, load M_{i+1}. After such a subsequence the robot is repositioned to start such a subsequence again et cetera. This leads to the following definition:

Definition 1 The sequence of robot moves

1. Unload M_i,
2. Travel (with the just unloaded part) from M_i to M_{i+1},
3. Load M_{i+1}

is called *(robot) activity* A_i for $i = 0, \ldots, m$.

It is not hard to see that not every sequence of activities constitutes a feasible sequence of operations for the robot to execute. For example, the robot cannot perform A_i immediately after completing A_i, since machine M_i is still empty.

Definition 2 An infinite sequence π of activities $A_0, \ldots, A_m$ is called a *feasible robot move sequence* if,

1. the robot never has to unload any empty machine,
2. the robot never has to load any loaded machine.

Since production efficiency in repetitive manufacturing environments is usually measured in terms of cycle times or (equivalently) throughput rates, we choose as objective to minimize cycle time. This yields that we are interested in the performance of the flowshop in a steady state, long run, situation. For practical purposes, it is not feasible to specify explicitly all the operations that the robot must perform in the long run. Instead, it is customary to prescribe some 'short' sequence of robot moves that the robot executes repeatedly, and we consequently restrict the analysis to cycle times that can be achieved when repeatedly executing certain classes of short sequences. In fact, all research presented in the open literature restricts the analysis to repetitively executing a finite robot move sequence. Since each machine must be loaded and unloaded when the flowshop is in operation, a short robot move sequence that is to be repeatedly executed must contain each activity at least once. Sequences in which each activity is executed exactly once form the simplest repeatable sequences:

Definition 3 A 1-unit cycle is a sequence of activities $A_0, \ldots, A_m$ in which each activity occurs exactly once and which constitutes a feasible robot move sequence when executed repeatedly.

Thus, each 1-unit cycle is a permutation of the activities $A_0, A_1, \ldots, A_m$. Interestingly, the converse statement is, in general, also true, i.e.

Theorem 1 (Lieberman and Turksen [1981], Sethi et al. [1992]) Every permutation of the activities $A_0, A_1, \ldots, A_m$ is a 1-unit cycle.

Since we are interested in the long run behavior of the flowshop, we assume in the remainder that we are free to specify the initial loaded/unloaded state of the machines. This yields specifically that we do not have to assume that the flowshop is initially empty.

To complete the problem statement, we continue the description of the flowshop. We require that the travel distances for the robot between machines M_i, M_j are given by means of a symmetric *distance matrix* D, whose elements δ_{ij} satisfy the *triangle equality*:

$$\delta_{ij} + \delta_{jk} = \delta_{ik}, \text{ for } 0 \leq i < j < k \leq m+1.$$

This equality models that the robot travels (or rotates) at constant speed along a trajectory. Further, there is a *loading and unloading time* ϵ_i for each machine $M_i, i = 0, \ldots, m+1$.

The processing requirements of the (identical) parts on machine $M_i, i = 1, \ldots, m$ are given by means of processing windows $[L_i, U_i]$: these requirements mean that each part must spend at least L_i time units and at most U_i time units on machine M_i. Such processing requirements naturally arise for instance in a manufacturing situation where the parts have to undergo some chemical treatment that may last neither too short nor too long (see e.g. Philips and Unger [1988], Lei [1993]). Further, the more common situation where a part may reside at a machine arbitrary long after it has been processed can be modelled by setting $U_i = \infty, i = 1, \ldots, m$. (Hall et al [1994], Sethi et al [1992], Crama and Van de Klundert [1994]). Finally, yet another special case arises when the parts are required to be unloaded as soon as they finish processing, see e.g. Levner et al. [1996]. This can be modelled by setting $L_i = U_i, i = 1, \ldots, m$.

A general problem description is now as follows:

Definition 4 *Robotic Flowshop Scheduling Problem* (RFSP):

INPUT : $D, \epsilon_i, [L_i, U_i]$ $(i = 0, \ldots, m+1)$, integer Z.

QUESTION : Is there a 1-unit cycle which when repeatedly executed yields a cycle time of at most Z.

In the remainder of the paper we will show the RFSP to be strongly NP-complete. Notice that proving NP-completeness for the identical parts case implies NP-completeness for the case were parts may have different processing requirements. To our knowledge, this

is the first NP-completeness proof that incorporates a realistic distance matrix. An earlier completeness proof of Lei and Wang [1989] assumed for example non-zero travel time between M_i and M_i. That the modelling of the distance matrix is of major importance, can be concluded from results of Crama and Van de Klundert [1994] and Levner et al. [1996] who provide polynomial algorithms for the cases where $U_i = +\infty$, and $L_i = U_i$ resp., that exploit the triangle equality property of the robot travel times.

To formalize the cycle time minization objective, we define

Definition 5 A *schedule* S is defined as a specification of starting times for each load and unload operation. More specifically, we denote by $S(l,i,t)$ ($S(u,i,t)$) the time at which the t-th loading (unloading) of machine M_i starts in schedule S ($i = 0,\ldots,m,\ t \in \mathbb{N}$).

The reader should notice that it not trivial to find a feasible schedule once the order in which the activities are to be executed is known, since the schedule has to respect the lower- and upperbounds of the processing windows. Notice also that this yields that not every 1-unit cycle is feasible for every problem instance. In addition, it is far from trivial to find a schedule with minimum cycle time once the 1-unit cycle is known. For this reason, researchers have commonly restricted the analysis to the special case where the robot executes a cyclic schedule, namely a so called 1-periodic schedule.

Definition 6 A *schedule* S is 1-periodic if there exists a constant C_S such that $S(l,i,t+1) - S(l,i,t) = C_S$ and $S(u,i,t+1) - S(u,i,t) = C_S$ for all $i = 0,\ldots,m+1, t \in \mathbb{N}$.

Obviously the cycle time of a 1-periodic schedule S equals C_S. Notice that the execution of a 1-periodic schedule forces the robot to repeat a 1-unit cycle, to be called $\pi(S)$. Without loss of generality, assume that the 1-unit cycle start with activity A_0. Then, S is feasible if the following relations are satisfied (Lei [1993]):

If A_{i-1} precedes A_i in $\pi(S)$, then

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

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

On the other hand, if A_i precedes A_{i-1} in $\pi(S)$, then

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

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

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

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

Furthermore, if A_k is the activity succeeding A_j in $\pi(S)$ then

$$S(l,j+1,t) + \epsilon_{j+1} + \delta_{j+1,k} \leq S(u,k,t), \tag{6}$$

and, if A_j is the last activity in $\pi(S)$, and A_k the first,

$$S(l,j+1,t) + \epsilon_{j+1} + \delta_{j+1,k} \leq S(u,k,t) + C_S. \tag{7}$$

As Lei [1993] observed, the optimal cycle time can be computed in polynomial time once π is known, since minimizing C_S subject to $(1) - (7)$ yields a linear programming problem. Now, for each π', let $C_{\pi'}$ be the minimum long run cycle time attainable by a schedule S satisfying $(1)-(7)$ such that $\pi(S) = \pi'$. Then, RFSP boils down to determining whether there is a 1-unit cycle π for which $C_\pi \leq Z$.

We finish this section by reviewing the literature in which this problem and closely related ones are addressed. The problem was introduced by Philips and Unger [1976]. They formulate the problem as an integer linear program, and solve some instances using standard software. Lieberman and Turksen [1981] formulate several related problems, e.g. problems in which there is more than one robot or problems in which the cell is not restricted to be a flowshop. Song et al. [1993] propose heuristics to find the optimal k-unit feasible robot move sequence for the no-wait version of this problem. In Lei [1993], the problem of minimizing the cycle time for a given permutation of the activities is shown to be solvable in $O(m^2 \log m \log B)$, where B depends linearly on the input parameters. Lei and Wang [1994], Armstrong, Lei and Gu [1994] and Hanen and Munier [1994] discuss branch & bound procedures for RFSI and alike. In Lei, Armstrong and Gu [1993], and in Lei and Wang [1991], heuristic procedures for a similar problem with multiple robots are given. For a more general overview of materials handling related scheduling problems in robotic cells we refer to Crama [1995] and van de Klundert [1996].

3 The NP-Completeness proof

Theorem 2 RFSP is strongly NP-Complete.

Proof. Membership in NP follows from $(1) - (7)$ (see e.g. Lei [1993]). We show its completeness by giving a reduction from the Bin Packing Problem to RFSP.

Bin Packing :

INPUT : Finite set $V = \{v_1, \ldots, v_q\}$ of items, a size $s(v_i) \in \mathbb{Z}^+$ for each $v_i, i = 1, \ldots, q$, positive integer $K \leq q$ and a positive integer B.

QUESTION : Is there a partition of V into disjoint sets $V_1, \ldots, V_K$ such that the sum of the sizes of the items in each V_i is B or less?

Consider an instance of the Bin Packing problem and assume without loss of generality that $s(v_i) \leq B$ for $i = 1, \ldots, q$. We construct an instance of RFSP as follows. There are $m = 2q + 1 + 2K$ machines. The processing windows of the machines $M_1, \ldots, M_K$ and machines $M_{2q+K+2}, \ldots, M_{2q+2K}$ are $[(4q + 2K + 3)B, (4q + 2K + 3)B]$. The processing window of $M_{2q+2K+1}$ is $[(4q + 2K + 2)B, (4q + 2K + 2)B]$. The windows of the machines $M_{K+2i+1}, i = 0, \ldots, q$ are $[0, +\infty]$. Finally the windows of the machines $M_{K+2i}, i = 1, \ldots, q$ are $[s(v_i), s(v_i)]$. The loading and unloading times ϵ_i are all equal to 0. The travel time between two adjacent machines equals B.

The following claim will be useful in the remainder of the proof :

Claim 1 For all $0 \leq i < K$, if A_i precedes A_{i+1} then activities $A_j, j \geq 2q + K + i + 2$ cannot be performed between the execution of A_i and A_{i+1}.

Proof. The processing window of machine M_{i+1} is $[(4q+2K+3)B, (4q+2K+3)B]$, and hence $(4q+2K+3)B$ time units after loading the machine it must be unloaded. To perform an activity with index at least $2q+K+i+2$ the robot must travel to machines with index at least $2q + K + i + 3$. Travelling to machine with index at least $2q + K + i + 3$ and back *between* the execution of A_i and A_{i+1}, requires at least $2 \times (2q+K+2)B > (4q+2K+3)B$ time, causing the schedule to be infeasible. ■

Claim 2 There is a 1-periodic schedule with cycle time $(4q+2K+4)KB+(4q+3K+4)B$ if and only if the bin packing instance is a yes instance.

Proof. To prove the claim, we first show that the activities

$$A_0, \ldots, A_K, A_{2q+K+2}, \ldots, A_{2q+2K+1}$$

must be in some specific order in every permutation that denotes a solution with the desired cycle time. Without loss of generality we may assume A_0 to be the first activity in the permutation. We claim activities A_0 to A_K are in order of their index in every feasible solution. Suppose not: let $i \in \{1, \ldots, K-1\}$ be the smallest index for which A_{i+1} precedes A_i (notice that $i \neq 0$). We consider two cases :

1. $A_{2q+2K+1}$ is scheduled before A_i. Let $A_j, j \leq i$ be the activity such that $A_{2q+2K+1}$ takes place between A_{j-1} and A_j. It follows from the Claim 1 that the schedule is infeasible.

2. $A_{2q+2K+1}$ is scheduled after A_i. This implies that between the execution of A_i in some iteration of the schedule and the execution of A_{i+1} in the next execution of the schedule the robot must perform $A_{2q+2K+1}$ and A_0 in that order. It follows again that the total travel time between A_i and A_{i+1} causes the schedule to be infeasible.

Thus activities $A_0, \ldots, A_K$ must indeed be in increasing order of their index. It is also straightforward to check that $A_{2q+K+1}, A_{2q+K+2}, \ldots, A_{2q+2K+1}$ must occur in this order in any feasible schedule (if $A_{2q+K+i+1}$ is performed before A_{2q+K+i}, then the travel time from the machine $M_{2q+K+i+1}$ to M_0 and back exceeds the processing window of $M_{2q+2K+i+1}$).

Let A_0 start at time 0. Considering the processing windows of machines $M_1, \ldots, M_K$ we can derive that the robot cannot start performing activity $A_i, i = 1, \ldots, K$ before time $(4q+2K+3)iB+iB$. More specifically, A_K cannot be started before $(4q+2K+3)KB+KB$. It can also be concluded from Claim 1 that $A_{2q+2K+1}$ cannot take place before A_K since otherwise the schedule would be infeasible. Combining these two observations leads to the conclusion that the total cycle time must be at least $(4q+2K+3)KB+KB+(2q+K+1)B+B+(2q+2K+2)B = (4q+2K+4)KB+(4q+3K+4)B$. Thus we have proved that this quantity (see Claim 2) is a lowerbound on the cycle time.

Claim 1 implies that $A_{2q+K+1+i}$ cannot precede A_i in any solution, for $i = 1, \ldots, K$. We are now going to show that A_{2q+2K} must precede A_K in every schedule having the desired cycle time. First of all, observe that $A_{2q+2K+1}$ cannot precede A_K. Hence if A_{2q+2K} is scheduled after A_K, it is either scheduled between A_K and $A_{2q+2K+1}$ or after $A_{2q+2K+1}$. In the latter case, the schedule was shown above to be infeasible. Thus A_{2q+2K} is scheduled between A_K and $A_{2q+2K+1}$. Now the total cycle time is at least the sum of the following time periods :

1. The interval from A_0 to the start of A_K, execution of A_K and travel time to machine M_{2q+2K} : taking time $(4q+2K+4)KB+B+(2q+2K-K)B$,

2. perform A_{2q+2K}, and wait or do something else until machine $M_{2q+2K+1}$ has finished processing : $B+(4q+2K+2)B$,

3. Unload $M_{2q+2K+1}$, bring the part to the output device and travel back to the input device to start the next execution of A_0 : $B+(2q+2K+3)B$.

This would result in a total cycle time of at least $(4q+2K+4)KB+B+(2q+2K-K)B+B+(4q+2K+2)B+B+(2q+2K+3)B > (4q+2K+4)KB+(4q+3K+4)B$. Now, since Claim 1 implies that A_{2q+2K} cannot precede A_{K-1}, we know that activities A_{K-1}, A_K and A_{2q+2K} are in the order A_{K-1}, A_{2q+2K}, A_K. It is easy to check that $A_{2q+2K-1}$ cannot be scheduled between A_{K-1} and A_K too. Moreover, since $A_{2q+2K-1}$ cannot be scheduled before A_{K-2}, as results from Claim 1, and cannot be scheduled after A_{2q+2K}, it must be scheduled between A_{K-2} and A_{K-1}. An inductive argument then establishes that

$A_{2q+2K-i}$ takes place between A_{K-i-1} and A_{K-i}. Hence we conclude that in any schedule that achieves the desired cycle time, the activities $A_0, \ldots, A_K, A_{2q+2K+1}, \ldots, A_{2q+2K+1}$ must be performed in the order

$$A_0, A_{2q+K+1}, A_1, A_{2q+K+2}, \ldots, A_K, A_{2q+2K+1}.$$

The remainder of the proof is now to show how the other activities must be plugged in, so that a schedule with the desired cycle time is obtained if one exists, and that such a schedule exists if and only if the bin packing instance is a yes instance.

We make three observations :

1. A_{K+2i-1} and $A_{K+2i}, i = 1, \ldots, q$ must always occur consecutively in every feasible permutation, since $s(v_i) \leq B$.

2. All the trajectories traveled between the end of A_i and the start of $A_{i+1}, i = 0, \ldots, K-1$, can be traveled only twice, since otherwise the processing window of M_{i+1} is violated. This implies that we cannot schedule any activities between A_{2q+K+i} and A_i for $i = 1, \ldots, K$.

3. None of the activities A_i with index $K+1 \leq i \leq K+2q$ can be scheduled after $A_{2q+2K+1}$ since otherwise the cycle time will be too large.

Together, these three observations imply that we must schedule $K+1$ sets of pairs of consecutive activities between A_i and $A_{K+2q+i+1}$, for $i = 0, \ldots, K$. The total travel time between loading M_{i+1} in A_i and unloading M_{i+1} in $A_{i+1}, i = 0, \ldots, K-1$ amounts $(4q+2K+2)B$. In view of the processing windows, this leaves us B time to perform pairs of activities A_{K+2i+1}, A_{K+2i+2}. Between any such pair of activities the robot must wait $s(v_{i+1})$ time. Furthermore, we cannot schedule any activities after the execution of A_K, because of the processing window of $M_{2q+2K+1}$. Thus, a 1-periodic schedule with the desired cycle time exists if and only if the numbers $s(v_i)$ can be partitioned into K sets each having weight no more than B. This proves Claim 2. ■

Instead of focusing directly on schedules, we have in this paper attempted to make clear the distinction between 1-unit cycles and 1-periodic schedules. Notice now that, in the instances created in the reduction, only 1-periodic schedules can lead to cycle times equal to the lowerbound specified in Claim 2, which is a requirement for a yes-instance. We conclude that not only the problem of finding the optimal 1-periodic schedule, but even our more general statement of RFSP is strongly NP-complete. ■

In conclusion, it is worth mentioning that the minimum cycle time attainable by a 1-unit cycle can be strictly larger than the minimum cycle time attainable by any robot move sequence (Lei [1995]). Very little is known however about the cycle times of non 1-unit cycles, and the reduction in cycle time that can be attained through executing

more complex robot move sequences.

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* (1995) 237-259. 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.*

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

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

L. Lei, Determining the optimal starting times in a cyclic schedule with a given route, *Computers and Operations Research*, Vol. 20, No. 8, (1993) 807-816.

L. Lei, 1995, private communication.

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

L. Lei and T.J. Wang, Determining optimal cyclic hoist schedules in a single-hoist electroplating line, *IIE Transactions* Vol. 26, No. 2 (1994) 25-33.

E. Levner, V.B. Kats, and V.E. Levit, An improved algorithm for a cyclic robotic scheduling problem. *Proceedings of the international workshop on intelligent scheduling of robots and flexible manufacturing systems*, Center for Technological Education Holon, Israel, (1996) 129-141.

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

L.W. Philips and P.S. Unger, Mathematical programming solution of a hoist scheduling program, AIIE Transactions, Vol. 8, No. 2, (1976) pp. 219-225.

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.