An Optimization-Based Heuristic for the Robotic Cell Problem

Jacques Carlier, Mohamed Haouari, Mohamed Kharbeche, Aziz Moukrim

An Optimization-Based Heuristic for the Robotic Cell Problem

Jacques Carlier[1], Mohamed Haouari[2,3],
Mohamed Kharbeche[4], Aziz Moukrim[1]

(1) UMR CNRS 6599 Heudiasyc, Centre de Recherches de Royallieu,
Université de Technologie de Compiègne, France.

(2) Department of Industrial Engineering, Faculty of Engineering,
Ozyegin University, Istanbul, Turkey.

(3) Princess Fatimah Alnijris' Research Chair for AMT, College of Engineering,
King Saud University, Saudi Arabia,

(4) ROI - Combinatorial Optimization Research Group,
Ecole Polytechnique de Tunisie, 2078 La Marsa, Tunisie.

Abstract

This study investigates an optimization-based heuristic for the *Robotic Cell Problem.* This problem arises in automated cells and is a complex flow shop problem with a single transportation robot and a blocking constraint. We propose an approximate decomposition algorithm. The proposed approach breaks the problem into two scheduling problems that are solved sequentially: a flow shop problem with additional constraints (blocking and transportation times) and a single machine problem with precedence constraints, time lags, and setup times. For each of these problems, we propose an exact branch-and-bound algorithm. Also, we describe a genetic algorithm that includes, as a mutation operator, a local search procedure. We report the results of a computational study that provides evidence that the proposed optimization-based approach delivers high-quality solutions and consistently outperforms the genetic algorithm. However, the genetic algorithm delivers reasonably good solutions while requiring significantly shorter CPU times.

Keywords : Flow Shop, Robotic Cell, Blocking, Branch-and-bound, Genetic Algorithm.

1 Introduction

The *Robotic Cell Problem* (*RCP*) is a generalization of the classical permutation flow shop problem. It may be formulated as follows. Given a job set $J = \{1, 2, ..., n\}$ where each

job has to be processed nonpreemptively on m machines M_1, M_2,..., M_m in that order. The processing time of job j ($j = 1, ..., n$) on machine M_i ($i = 1, ..., m$) is p_{ij}. At time $t = 0$, all jobs are available at an input device denoted by M_0. After completion, each job must be taken from M_m to an output device that is denoted by M_{m+1} (for convenience, we set $p_{m+1,j} = 0, \forall j \in J$). The transfer of a job $j \in J$ from M_i to M_{i+1} ($i = 0, ..., m$) is performed by means of a single robot. An empty or a loaded move of the robot from M_i to M_h ($i, h = 0, ..., m+1$) takes τ_{ih} units of time. The machines have neither input nor output buffering facilities. Consequently, after processing a job j on machine M_i ($i = 1, ..., m$), this latter remains blocked until the robot picks up j and transfers it to the following machine M_{i+1}. Such a move could only be performed if machine M_{i+1} is free (that is, no job is being processed by or waiting at M_{i+1}). At any time, each machine can process at most one job and each job can be processed on at most one machine. Moreover, the robot can transfer at most one job at any time. The problem is to find a processing order of the n jobs, the same for each machine (because of the blocking constraint, passing is not possible), such that the time $C_{\max}$ at which all the jobs are completed (makespan) is minimized.

The particular case of the robotic cell problem where the transportation times are negligible reduces to the much studied flow shop scheduling problem with blocking (Pinedo, 2008). Hall and Sriskandarajah (1996) prove that this latter problem is strongly $\mathcal{NP}$-hard for $m \geq 3$. Consequently, the RCP is strongly $\mathcal{NP}$-hard for $m \geq 3$ as well.

Our motivation for the investigation of the robotic cell problem stems from its practical relevance to Flexible Manufacturing Systems (FMSs), which are highly automated production systems capable of producing a wide variety of job types. As pointed out in Blazewicz et al. (1991), one of the most difficult operational problems in FMSs is the development of effective schedules considering jobs, machines and transportation devices in order to provide a proper coordination of the production sequencing and time allocation of all required resources. Blocking constraints arise for example in the manufacturing process or the chemical industry where some characteristics (such as temperature) of the material requires that each job must wait on the machine before being processed on the next machine. The RCP is a specific problem in FMSs. Only very special cases can be solved in polynomial-time and various cases with a single robot are already $\mathcal{NP}$-hard (see Knust (1999)).

The main contribution of this paper, is to propose an approximate decomposition algorithm for the RCP that requires sequentially solving two scheduling problems. Each of these problems is solved *exactly*. More precisely, we propose the following two-phase solution strategy:

- **Phase 1**: We solve a relaxation of RCP that is derived by *partially* relaxing the robot capacity constraints. More precisely, we assume that after transferring a job j from M_i to M_{i+1} ($i = 1, ..., m$), the robot is *always* immediately available to perform an empty move from M_{i+1} to M_{i-1} and pick up the next scheduled job k when it becomes ready. Moreover, we assume that after finishing a job j on the last machine M_m, the robot can immediately transfer this job to the output device. Finally, we assume that

the first scheduled job will be processed without waiting at any machine. With these assumptions, we obtain a flow shop problem with blocking and transportation times. We shall provide a formal description of this problem in Section 3. The solution of this relaxed problem (hereafter, denoted by $F|Block, t_k|C_{\max}$) yields a solution which consists of a permutation σ of the jobs with a corresponding completion time $C^1_{\max}(\sigma)$.

- **Phase 2**: Given the permutation σ that is derived in Phase 1, we determine the sequence of robot moves (including both empty as well as loaded moves) that minimizes the maximum completion time (makespan). In Section 4, we show that this amounts to solving a single machine problem with side-constraints. In doing so, we obtain a completion time $C^2_{\max}(\sigma)$.

Clearly, since the flow shop problem that is defined in Phase 1 is a relaxation of the RCP then the optimal makespan $C^1_{\max}(\sigma)$ is a valid lower bound on the optimal makespan of the RCP. Moreover, the optimal makespan of the solution of the single machine problem that is defined in Phase 2 is an upper bound on the optimal makespan of the RCP.

The remainder of this paper is organized as follows. In Section 2, we review the literature pertaining to job scheduling and transportation planning in flexible manufacturing systems. In Section 3, we present several lower bounds for $F|Block, t_k|C_{\max}$ and we describe an exact branch-and-bound algorithm for solving this latter problem. In Section 4, we describe an optimization algorithm for sequencing the robot moves. In Section 5, we describe a genetic algorithm. The empirical performance of the proposed approach is assessed in Section 6 where we present the results of a comprehensive computational study. Finally, we provide in Section 7 some concluding remarks and we outline topics for future investigations.

2 Literature review on scheduling and transportation planning in FMSs

The impressively rapid spread of robotic cells in manufacturing systems that occurred during the last two decades has prompted the emergence of many new scheduling problems. We refer to the excellent book of Dawande et al. (2007) for an up-to-date and comprehensive review of sequencing and scheduling problems arising in robotic cells. Actually, almost all previously investigated robotic cell scheduling problems deal with *cyclic scheduling* problems with machines producing a family of similar parts, in a steady-state. We refer to Dawande et al. (2005) for a classification scheme of these challenging problems. However, to the best of our knowledge, the multiple-part-type robotic cell problem that is addressed in this paper has never been investigated before in the foregoing very general form. Indeed, multiple-part type problems that have been addressed so far are often defined as follows. We are given a minimal part set (MPS) that includes p different part-types to be produced. Each type k ($k = 1, ..., p$) includes d_k similar parts. The MPS requires to be scheduled *repetitively*

using p repetitions of a one-unit robot move cycle with the objective of maximizing the throughput rate, or equivalently, minimizing the cycle time (for example see Sriskandarajah et al. (1998)).

Furthermore, although there is a vast literature pertaining to integrated job and vehicle scheduling (see Ganesharajah et al. (1998)), there are few papers which consider flow shop problem with blocking to minimize makespan (see Hall and Sriskandarajah (1996)). Mascis and Pacciarelli (2002) study the job shop problem with blocking. They formulate it by means of a generalization of the disjunctive graph of Roy and Sussman (1964) and they show that several properties used to design heuristic procedures do not hold in the blocking case. Reddi and Ramamoorthy (1972) show that the problem $F2|Block|C_{\max}$ could be reduced to a special case of the traveling salesman problem that can be solved in polynomial time by the Gilmore and Gomory (1964) algorithm. Dutta and Cunningham (1975) propose dynamic programming procedures to solve this problem with limited buffer. Levner (1969) presents branch-and-bound algorithms for solving the flow shop problem with blocking. Mc Cormick et al. (1989) study a flow shop scheduling problem in an assembly line where the problem with limited buffers can be studied as a blocking one (all machines have no intermediate buffers). Abadi et al. (2000) propose a heuristic for minimizing the cycle time in order to establish a connection between the no-wait flow Shop problem and flow shop with blocking. This approach has been used by Caraffa et al. (2001) for calculating the value of makespan for a given sequence of the jobs. Ronconi (2005) proposes a branch-and-bound algorithm using a lower bound that exploits the blocking feature. The author shows that the obtained bounding scheme outperforms the lower bounds proposed in Ronconi and Armentano (2001). Recently, Grabowski and Pempera (2007) propose two new heuristic algorithms to minimize the makespan in a flow shop problem with blocking based on a tabu search approach. Lee and Chen (2001) and Hurink and Knust (2001) studied flow shop scheduling with explicit transportation capacity and transportation times. Moreover, they assumed an unlimited buffer space between the machines and negligeable empty moving times. The literature pertaining to other scheduling models with transportation aspects is summarized in Crama et al. (2000). In addition, Agnetis (2000) and Agnetis and Pacciarelli (2000) investigate the complexity of a *no-wait* flow shop problem in which one robot is used to move the parts from a machine to the next, as well as between the machines and the input/output devices. In this model, jobs are not allowed to wait neither on a machine nor on the robot.

Not surprisingly, most robotic cell scheduling problems are intractable. In particular, Hall and Sriskandarajah (1996) prove that the robotic cell problem is strongly $\mathcal{NP}$-hard for $m \geq 3$. However, an $O(n^4)$ algorithm that solves this problem in two-machine cells is provided in Hall et al. (1997). Aneja and Kamoun (1999) improve this complexity to $O(n \log n)$.

3 Solution of the flow shop problem with blocking and transportation delays

In this section, we are concerned with the exact solution of the flow shop problem with blocking and transportation times $(F|Block, t_k|C_{\max})$ that requires to be solved in Phase 1 after *partially* relaxing the constraints on the robot capacity. Clearly, since this problem generalizes several flow shop problems that are strongly $\mathcal{NP}$-hard for $m \geq 3$, then it is itself strongly $\mathcal{NP}$-hard for $m \geq 3$ (Hall and Sriskandarajah (1996)). In the sequel, we shall prove that the problem is solvable in $O(n^2)$ for $m = 2$.

Given a permutation of the n jobs $\sigma = (\sigma(1), \sigma(2), ..., \sigma(n))$, we denote by $t_{i,\sigma(j)}$ the starting time of job $\sigma(j)$ $(j = 1, ..., n)$ on machine M_i $(i = 1, ..., m)$. The robot has to transfer job $\sigma(j)$ $(j = 2, ..., n)$, after its processing on machine M_{i-1} $(i = 2, ..., m)$, from machine M_{i-1} to machine M_i. Hence, we have $t_{i,\sigma(j)} \geq t_{i-1,\sigma(j)} + p_{i-1,\sigma(j)} + \tau_{i-1,i}$. Also, after unloading job $\sigma(j-1)$ on machine M_{i+1}, the robot has to move to machine M_{i-1} to upload job $\sigma(j)$ and transfer it to machine M_i. Hence, we have $t_{i,\sigma(j)} \geq t_{i+1,\sigma(j-1)} + \tau_{i+1,i-1} + \tau_{i-1,i}$. Therefore, the starting times can be computed through the following recursive equations:

$$\begin{array}{lcll} t_{0,\sigma(j)} & = & 0 & j = 1, ..., n, \\ t_{i,\sigma(1)} & = & t_{i-1,\sigma(1)} + p_{i-1,\sigma(1)} + \tau_{i-1,i} & i = 1, ..., m+1, \\ t_{1,\sigma(j)} & = & t_{2,\sigma(j-1)} + \tau_{20} + \tau_{01} & j = 2, ..., n, \\ t_{i,\sigma(j)} & = & \max(t_{i-1,\sigma(j)} + p_{i-1,\sigma(j)}, t_{i+1,\sigma(j-1)} + \tau_{i+1,i-1}) + \tau_{i-1,i} & j = 2, ..., n,\ i = 2, ..., m, \\ t_{m+1,\sigma(j)} & = & t_{m,\sigma(j)} + p_{m,\sigma(j)} + \tau_{m,m+1} & j = 2, ..., n. \end{array} \tag{1}$$

Moreover, the completion time $C_{i,\sigma(j)}$ of job $\sigma(j)$ on machine M_i is

$$C_{i,\sigma(j)} = t_{i,\sigma(j)} + p_{i,\sigma(j)} \quad \forall i = 1, ..., m+1, j = 1, ..., n \tag{2}$$

and the makespan is $C_{m+1,\sigma(n)}$.

The $F|Block, t_k|C_{\max}$ requires finding a permutation σ^* such that $C_{m+1,\sigma^*(n)}$ is minimal.

3.1 Lower bounds

In this section we describe one-machine as well as two-machine relaxations for $F|Block, t_k|C_{\max}$. Consequently, the derived lower bounds are also valid for the RCP as well.

3.1.1 One-machine based lower bounds

As a consequence of the transportation delays, the minimum elapsed time on machine M_i between the completion time of a job j and the starting time of a following job k is

$$\delta_i = \tau_{i,i+1} + \tau_{i+1,i-1} + \tau_{i-1,i} \quad \forall i = 1, ..., m \tag{3}$$

Also, by setting for each job $j \in J$ and each machine M_i $(i = 1, ..., m)$:

- a head (or, release date) $r_{ij} = \sum_{k=1}^{i-1} p_{kj} + \sum_{k=0}^{i-1} \tau_{k,k+1}$ if $i > 1$ and $r_{1j} = \tau_{01}$,
- a tail (or, delivery time) $q_{ij} = \sum_{k=i+1}^{m} p_{kj} + \sum_{k=i}^{m} \tau_{k,k+1}$ if $i < m$ and $q_{mj} = \tau_{m,m+1}$.

Hence, a simple $O(mn)$ lower bound is

$$LB_1 = \max_{1 \le i \le m} \{ \min_{1 \le j \le n} r_{ij} + \sum_{j=1}^{n} p_{ij} + (n-1)\delta_i + \min_{1 \le j \le n} q_{ij} \} \tag{4}$$

Remark 1 *A better bound can be derived by observing that if job k is scheduled immediately after job j, then the minimum elapsed time on machine M_i $(i = 2, ..., m)$ between the completion of j and the starting of a job k is given by*

$$s_{ijk} = \max(p_{ij} + \tau_{i,i+1} + \tau_{i+1,i-1}, p_{i-1,k} + \tau_{i,i-2} + \tau_{i-2,i-1}) - p_{ij} + \tau_{i-1,i} \quad \forall i = 2, ..., m \tag{5}$$

Proof. Let t_{ij} denote the starting time of job j on machine M_i. At $t_{ij} + p_{ij}$, job j is ready to be transferred to M_{i+1} and could start processing on machine M_{i+1}at $t_{ij} + p_{ij} + \tau_{i,i+1}$. Thus, the next job (say, k) could not be transferred from M_{i-1} to M_i before time $t_{ij} + p_{ij} + \tau_{i,i+1} + \tau_{i+1,i-1}$. Moreover, job k cannot be transferred before time $t_{ij} + \tau_{i,i-2} + \tau_{i-2,i-1} + p_{i-1,k}$ which is the earliest finish time of job k on machine M_{i-1}. Therefore, the earliest start time of k on M_i is $t_{ij} + \max(p_{ij} + \tau_{i,i+1} + \tau_{i+1,i-1},\ p_{i-1,k} + \tau_{i,i-2} + \tau_{i-2,i-1}) + \tau_{i-1,i}$. Thus, the result follows. ■

Now, define $\delta_{ij} = \min_{k \ne j} \{ s_{ijk} \}$, $\forall i = 2, ..., m, j = 1, ..., n$ ($\equiv$ minimum elapsed time after completion of j on M_i), $\delta^i_{[k]} = k^{th}$ smallest value of δ_{ij} $(j = 1, ..., n)$. Then we get the lower bound

$$LB_2 = \max_{2 \le i \le m} \{ \min_{1 \le j \le n} r_{ij} + \sum_{j=1}^{n} p_{ij} + \sum_{k=1}^{n-1} \delta^i_{[k]} + \min_{1 \le j \le n} q_{ij} \} \tag{6}$$

LB_2 can be computed in $O(mn^2)$ time.

Moreover, a valid relaxation is a one-machine problem with heads, tails, and setup times $1 \mid r_j, q_j, s_{jk} \mid C_{\max}$. A relaxation of this problem is a $1 \mid r_j, q_j \mid C_{\max}$ obtained by setting

$r'_j = r_{ij}$, $p'_j = p_j + \delta_{ij}$, and $q'_j = q_{ij} - \delta_{ij}$ for $j = 1, ..., n$. Hence, a third lower bound is obtained by allowing preemptive schedules. An optimal preemptive schedule is obtained in $O(n \log n)$ time by using the list schedule associated with the longest tail priority dispatching rule. To build it, we schedule at each time t the available operation with maximal tail. For each machine M_i $(i = 1, ..., m)$, let $LB3_i$ denote the makespan of the corresponding optimal preemptive schedule. Then a valid $O(mn^2)$-time lower bound is

$$LB3 = \max_{2 \leq i \leq m} LB3_i \tag{7}$$

A further relaxation of $1 \mid r_j, q_j, s_{jk} \mid C_{\max}$ is obtained by setting all heads and tails to $\min_{j \in J}\{r_j\}$ and $\min_{j \in J}\{q_j\}$, respectively. The resulting relaxation is equivalent to finding a shortest Hamiltonian path in a directed complete graph, where the nodes represent the jobs and the distance matrix is (s_{ijk}). We can transform this problem into an equivalent asymmetric traveling salesman problem (ATSP) by adding to the graph a dummy node 0 and dummy zero-cost arcs $(j, 0)$ and $(0, j)$, for $j = 1, ..., n$. For a given machine M_i, let z^i_{ATSP} denote the value of the shortest cycle. Since solving this relaxed problem is $\mathcal{NP}$-hard, we compute a lower bound on z^i_{ATSP}. In our implementation, we derive a tight lower bound LB^i_{ATSP} by solving the linear relaxation of a polynomial-size mixed-integer programming formulation that is defined as follows.

Define, the underlying complete digraph $\vec{G} = (\vec{V}, \vec{A})$ where the node set $\vec{V} \equiv J \cup \{0\}$ and the cost of arc $(j, k) \in \vec{A}$ is s_{ijk}. The decision variables are

- $x_{jk} = 1$ if node k is visited immediately after node j, and 0 otherwise, $\forall (j, k) \in \vec{A}$,
- u_j : position in which node j is visited, $j = 1, ..., n$.

The formulation is

$$\text{Minimize} \sum_{(j,k) \in \vec{A}} s_{ijk} x_{jk} \tag{8}$$

subject to:

$$\sum_{k=0,\ k \neq j}^{n} x_{jk} = 1, \qquad j = 0, ..., n, \tag{9}$$

$$\sum_{j=0,\ k \neq j}^{n} x_{jk} = 1, \qquad k = 0, ..., n, \tag{10}$$

$$n x_{jk} + (n-2) x_{kj} + u_j - u_k \leq n - 1, \qquad \forall j, k \geq 1 \quad j \neq k, \tag{11}$$

$$1 + (1 - x_{0j}) + (n-2) x_{j0} \leq u_j \leq n - (n-2) x_{0j} - (1 - x_{j0}), \quad \forall j \geq 1, \tag{12}$$

$$u_j \geq 0, \qquad \forall\ j = 1...n, \tag{13}$$

$$x_{jk} \in \{0,\ 1\}, \qquad \forall j, k \geq 1 \quad j \neq k. \tag{14}$$

The objective function (8) minimizes the total distance (or, setup times). Constraints (9) and (10) require that each node has exactly one successor and predecessor, respectively. Constraints (11)-(13) are the subtour elimination constraints. These constraints are often referred to as *lifted Miller-Tucker-Zemlin subtour elimination constraints*. Finally, Constraints (14) enforce binary restrictions on the x-variables. Model (8)-(14) includes $O(n^2)$ constraints, $O(n^2)$ binary variables, and $O(n)$ continuous variables. This polynomial-size ATSP formulation has been proposed by Desrochers and Laporte (1991) and is an enhanced version of a simpler formulation that was first derived by Miller et al. (1960).

Let LB^i_{ATSP} denote the value of the linear relaxation of Model (8)-(14). We obtain a fourth lower bound

$$LB4 = \max_{2\leq i\leq m}\{\min_{1\leq j\leq n} r_{ij} + LB^i_{ATSP} + \min_{1\leq j\leq n} q_{ij}\} \tag{15}$$

In order to speed-up $LB4$, we can reduce the LP size by relaxing the subtour elimination constraints (11)-(13). Hence, the resulting relaxation is a linear assignment problem. We denote by $LB4^*$ the value of the corresponding bound. Since, the linear assignment problem can be solved using the Hungarian algorithm in $O(n^3)$ time, then $LB4^*$ can be computed in $O(mn^3)$ time.

3.1.2 A two-machine based lower bound

First, we consider the special case where $m = 2$. We shall prove that the problem is solvable in polynomial time. Given a permutation σ of the n jobs together with the associated starting times $t_{i,\sigma(j)}$ ($i = 1, 2$ and $j = 1, ..., n$). We denote by $C_{\max}$ the corresponding makespan. The time interval $[0, C_{\max}]$ can be partitioned into $2n + 1$ sub-intervals I_1, J_1, I_2, J_2,..., I_n, J_n, I_{n+1} where

- $I_1 = [0, t_{2,\sigma(1)} - \tau_{12}]$,
- $I_j = [t_{2,\sigma(j-1)}, t_{2,\sigma(j)} - \tau_{12}]$ for $j = 2, ..., n$,
- $I_{n+1} = [t_{2,\sigma(n)}, C_{\max}]$,
- $J_j = [t_{2,\sigma(j)} - \tau_{12}, t_{2,\sigma(j)}]$ for $j = 1, ..., n$.

Figure 1 depicts the partition of $[0, C_{\max}]$ for a 4-job schedule.

Insert Figure 1 here

We observe that:

- During each interval $J_j = [t_{2,\sigma(j)} - \tau_{12}, t_{2,\sigma(j)}]$ ($j = 1, ..., n$) both machines are idle during τ_{12} units of time

- Before starting processing job $\sigma(j)$ $(j = 2, ..., n)$, machine M_1 remains idle during at least $(\tau_{20} + \tau_{01})$ units of time. Also, M_1 remains idle during τ_{01} units of time before processing job $\sigma(1)$
- After finishing processing job $\sigma(j)$ $(j = 1, ..., n-1)$, machine M_2 remains idle during at least $(\tau_{23} + \tau_{31})$ units of time. Also, M_2 remains idle during τ_{23} units of time after processing job $\sigma(n)$

Based on these observations, we may include the transportation times into the processing times by letting

$$p'_{1j} = p_{1j} + \tau_{20} + \tau_{01} \quad \text{for } j = 1, ..., n \tag{16}$$
$$p'_{2j} = p_{1j} + \tau_{23} + \tau_{31} \quad \text{for } j = 1, ..., n \tag{17}$$

and adding to the resulting makespan the constant $n\tau_{12} - \tau_{20} - \tau_{31}$. Moreover, we observe that M_1 remains blocked until M_2 becomes available for processing. Hence, it is instructive to see the two-machine relaxation in another way: as a two-machine flow shop problem (with modified processing times) and blocking. Actually, this latter problem is equivalent to the no-wait two-machine flow shop problem (it is easily realized that both problems have the same makespan for the same job sequence). Hence, the two-machine flow shop problem with blocking could be restated as a traveling salesman problem (see for example Pinedo (2008), p. 180) and solved in $O(n \log n)$-time (Vairaktarakis, 2003).

Consequently, if $m > 2$, then we can consider a pair of consecutive machines (M_i, M_{i+1}) $(i = 1, ..., m-1)$ and we relax the capacities of all the other machines. The resulting relaxation is a two-machine permutation flow shop with blocking and transportation problem subject to heads and tails, where for each job j is defined

- a head $\tilde{r}_{ij} = \sum_{k=1}^{i-1} p_{kj} + \sum_{k=0}^{i-2} \tau_{k,k+1}$ if $i > 1$ and $\tilde{r}_{1j} = 0$,
- a tail $\tilde{q}_{ij} = \sum_{k=i+2}^{m} p_{kj} + \sum_{k=i+2}^{m} \tau_{k,k+1}$ if $i < m-1$ and $\tilde{q}_{m-1,j} = 0$.

Let LB_5^i denote the optimal makespan. Then a valid lower bound is

$$LB_5 = \max_{1 \le i \le m-1} \{ \min_{1 \le j \le n} \tilde{r}_{ij} + LB_5^i + \min_{1 \le j \le n} \tilde{q}_{i,j} \} \tag{18}$$

LB_5 can be computed in $O(mn^2)$ time.

3.2 An exact branch-and-bound algorithm

With each node N at level $l \geq 1$ of the search tree we associate a subsequence $\sigma(N)$ represented by an ordered list of l jobs scheduled on the top of the global sequence. The root node corresponds to an empty list. With each node N is associated the following data:

$\overline{J}(N)$: Set of unscheduled jobs,

C_σ^i : Completion time of σ on machine M_i $(i = 1,\ ....,\ m)$,

$j(\sigma)$: index of the last scheduled job,

C_{max}^* : The current best upper bound on the optimal makespan. In our implementation, the value of C_{max}^* at the root node is provided by the genetic algorithm that is described in Section 5.

$LB(N)$: A lower bound on the optimal makespan of a sequence where the jobs of subset $\overline{J}$ are scheduled after the completion of subsequence $\sigma(N)$.

Given a node N, a child node N^+ is created by sequencing an unscheduled job $k \in \overline{J}(N)$ at the last position of $\sigma(N)$ (thus, $\sigma(N^+) = \sigma(N)k$). For computing a lower bound $LB(N^+)$ we can use any of the previously described lower bounding procedures. However, we should take into account the partial schedule corresponding to the sequence σ. For the one-machine based bounds $L_1, ..., L_4$ the required modifications are straightforward. We describe how to modify LB_5 in order to account for the partial sequence σ. The basic idea for computing $LB_5^i(N)$ $(i = 1, ..., m-1)$ is the following. Firstly, in order to account for the availability of machine M_{i+1} we append to the set of unscheduled jobs a dummy job 0 such that $p_{i0} = 0$ and $p_{i+1,0} = p_{i+1,j(\sigma)}$. In so doing and if job 0 is scheduled at the first position, then machine M_{i+1} would be unavailable during the time interval $[C_\sigma^{i+1} - p_{i+1,j(\sigma)}, C_\sigma^{i+1}]$. Since, the robot requires $(\tau_{i-1,i} + \tau_{i,i+1})$ units of time for transferring the dummy job from M_{i-1} to M_{i+1}, then a lower bound on the starting time a_i of the robot moves is given by (see Figure 2)

$$a_i = C_\sigma^{i+1} - p_{i+1,j(\sigma)} - \tau_{i,i+1} - \tau_{i-1,i}, \qquad i = 1, ..., m-1. \tag{19}$$

Finally, we compute $LB_5^i(N)$ for the subset $\overline{J}(N) \cup \{0\}$ and we set

$$LB_5(N) = \max_{1 \leq i \leq m-1} \{a_i + LB_5^i(N) + \min_{j \in \overline{J}(N)} \tilde{q}_{i,j}\}. \tag{20}$$

Insert Figure 2 here

For selecting the node to be branched, we select the node which has the smallest lower bound among the most recently created nodes.

3.2.1 Exploiting the problem symmetry

Given an instance I of $F|Blocking, t_k|C_{\max}$, we define the corresponding symmetric instance I^{-1}, the instance that is obtained by reversing the machine ordering. That is, each job $j \in J$ must be picked up from M_{m+1}, and first processed on M_m, then on M_{m-1},..., M_1, in that

order, and ultimately transferred to M_0. Let $\sigma = (\sigma(1), ..., \sigma(n))$ denote a permutation of the n jobs, we define the associated reverse permutation $\sigma^{-1} = (\sigma(n), ..., \sigma(1))$. Moreover, we denote the makespan of the solution of instance I that corresponds to σ by $C_{\max}(I, \sigma)$.

Proposition 1 *Given an instance I satisfying the following properties:*

(P1) $\tau_{i,i+1} = \tau_{k,k+1}\ \forall\ i, k = 0, ...m$ *and* $\tau_{i,i+2} = \tau_{k,k+2}\ \forall\ i, k = 0, ...m-1$
(P2) $\tau_{i,i+1} = \tau_{i+1,i}\ \forall\ i = 0, ...m$
Then, we have $C_{\max}(I, \sigma) = C_{\max}(I^{-1}, \sigma^{-1})$ for all σ.

Proof. First, we shall prove that the makespan under a given permutation schedule can be obtained through computing a critical path in a digraph. Consider the instance I and a permutation σ of the n jobs. Recall that $t_{i,\sigma(j)}$ denotes the starting time of job $\sigma(j)$ on M_i $(i = 1, ..., m+1)$. Therefore, $C_{\max}(I, \sigma) = t_{m+1,\sigma(n)}$.

We construct an associated acyclic digraph $G(I, \sigma) = (V, A)$ as follows. The node set is $V = \{(i, \sigma(j) : i = 1, ..., m+1,\ j = 1, ..., n\}$. The arc set includes two types of arcs:

- Type 1: There is an arc $(i+1, \sigma(j-1)) \rightarrow (i, \sigma(j))$ for $i = 1, ..., m,\ j = 2, ..., n$
- Type 2: There is an arc $(i, \sigma(j)) \rightarrow (i+1, \sigma(j))$ for $i = 1, ..., m,\ j = 1, ..., n$.

The arcs are assigned the following weights:

- An arc $(i+1, \sigma(j-1)) \rightarrow (i, \sigma(j))$ of Type 1 has a weight $\tau_{i+1,i-1} + \tau_{i-1,i}$
- An arc $(i, \sigma(j)) \rightarrow (i+1, \sigma(j))$ of Type 2 has a weight $p_{i,\sigma(j)} + \tau_{i,i+1}$. However, there is one exception: the weight of arc $(1, \sigma(1)) \rightarrow (2, \sigma(1))$ is $p_{1,\sigma(1)} + \tau_{01} + \tau_{12}$

For the sake of clarity, Figure 3a displays the graph that is associated with a 3-job 3-machine instance and $\sigma = (1, 2, 3)$, and where $\tau_{ij} = \delta\,|i - j|$ for $i, j = 0, ..., m+1$. Moreover, Figure 3b displays the graph that is associated with the reverse instance and the permutation $\sigma^{-1} = (3, 2, 1)$. We define $\bar{G}(I, \sigma)$ the graph that is obtained from $G(I, \sigma)$ by deleting arcs $(1, \sigma(1)) \rightarrow (2, \sigma(1))$ and $(m, \sigma(n)) \rightarrow (m+1, \sigma(n))$ as well as nodes $(1, \sigma(1))$ and $(m+1, \sigma(n))$. We observe, that $\bar{G}(I^{-1}, \sigma^{-1})$ can be derived from $\bar{G}(I, \sigma)$ by reversing all the arcs in $\bar{G}(I, \sigma)$. Consequently, the value of the longest path from node $(2, \sigma(1))$ to node $(m, \sigma(n))$ in $\bar{G}(I, \sigma)$ (and therefore in $G(I, \sigma)$) is equal to the value of the longest path from node $(m, \sigma(n))$ to node $(2, \sigma(1))$ in $\bar{G}(I^{-1}, \sigma^{-1})$ (and therefore in $G(I^{-1}, \sigma^{-1})$).

Define, $\mathcal{L}(\alpha, \beta)$ as the value of the longest path in the graph between nodes α and β. One could readily check the following simple facts.

F1 The value of the makespan is equal to the value of the longest path in $G(I, \sigma)$ between nodes $(1, \sigma(1))$ and $(m+1, \sigma(n))$. This fact stems from Equations (1) and the graph structure. Hence, $C_{\max}(I, \sigma) = \mathcal{L}((1, \sigma(1)), (m+1, \sigma(n)))$.

F2 We see that $\mathcal{L}((1,\sigma(1)),(m+1,\sigma(n))) = p_{1,\sigma(1)} + \tau_{01} + \tau_{12} + \mathcal{L}((2,\sigma(1)),(m,\sigma(n))) + p_{m,\sigma(n)} + \tau_{m,m+1}$. This stems from the fact that nodes $(2,\sigma(1))$ and $(m,\sigma(n))$ have exactly one predecessor and one successor, respectively.

Consequently, since I satisfies (P1) and (P2), then $C_{\max}(I,\sigma) = C_{\max}(I^{-1},\sigma^{-1})$ ■

Corollary 2 *If I satisfies (P1) and (P2) and if σ is an optimal permutation for instance I, then σ^{-1} is an optimal permutation for I^{-1}.*

Consequently, if I satisfies (P1) and (P2) and if the branch-and-bound algorithm fails in finding an optimal solution within a given time limit then we take advantage of the symmetry of $F|Blocking, t_k|C_{\max}$ by solving the symmetric problem (that is obtained by reversing the machine ordering).

4 Scheduling of the robot moves

In this section, we are concerned with scheduling the robot moves provided that the processing order of the jobs on the m machines is given. For the sake of alleviating the notation we shall assume that the jobs are indexed according to their processing order.

It is interesting to view the robot sequencing problem as a single machine scheduling problem with side constraints, where the machine represents the robot and the operations correspond to the robot *loaded* moves, respectively. More precisely, we assume that we have a set of operations $\Sigma = \{O_{ij} : i = 0,...,m,\ j = 1,...,n\}$ to be processed nonpreemptively on a single machine. An operation $O_{ij} \in \Sigma$ corresponds to a transfer of job j from M_i to M_{i+1}. The processing time of O_{ij} is $\tau_{i,i+1}$. In addition, there are two types of precedence constraints:

- Type 1: Because of the flow shop constraints, we have $O_{i-1,j} \prec O_{i,j}$ for $i = 1,...,m$ and $j = 1,...,n$ (where the notation "$u \prec v$" means that operation u precedes operation v). This constraint expresses that a job j cannot be transferred to machine M_{i+1} before being transferred to machine M_i.

- Type 2: Because of the blocking constraints, we have $O_{ij} \prec O_{i-1,j+1}$ for $i = 1,...,m$ and $j = 1,...,n-1$. This constraint enforces that job $j+1$ cannot be transferred to machine M_i before transferring job j from M_i to M_{i+1}.

Moreover, there is a minimal time lag $l_{i-1,j} \equiv p_{ij}$ between the completion time of operation $O_{i-1,j}$ and the start time of operation O_{ij} ($i = 1,...,m$ and $j = 1,...,n$). This time lag corresponds to the processing time of job j on machine M_{i+1} ($i = 0,...,m$). Finally, if two operations $O_{i-1,j}$ and O_{hk} are processed consecutively (with $j \neq k$) then there is a nonnegative setup time $\gamma_{ih} \equiv \tau_{ih}$ between these two operations. This setup time corresponds to an

empty positioning move from M_i to M_h. Thus, optimizing the robot moves requires minimizing the makespan on a single machine with precedence constraints, time-lags, and setup times. Hurink and Knust (2002) investigated this problem in the case where the precedence constraints stem from a job shop environment. They proved that the problem is $\mathcal{NP}$-hard and proposed a tabu search algorithm. However, the complexity of the problem when the precedence constraints stem from a flow shop environment remains an open question. In the sequel, we describe an exact branch-and-bound algorithm for solving this latter problem.

Given an optimal permutation $\sigma = (1, ..., n)$ of the n jobs, we construct an associated weighted digraph $\hat{G} = (\hat{V}, \hat{A})$ where each node $u \in \hat{V}$ corresponds to an operation $O_{ij} \in \Sigma$, respectively. An arc (u, v) that connects node $u \equiv (\alpha_1, \beta_1) \in \hat{V}$ to node $w \equiv (\alpha_2, \beta_2) \in \hat{V}$ is defined if and only if there is a precedence constraint $O_{\alpha_1\beta_1} \prec O_{\alpha_2\beta_2}$. That is,

(*i*) $\alpha_1 = \alpha_2 - 1$ and $\beta_1 = \beta_2$, the weight of this arc is $c(u, v) = \tau_{\alpha_2-1,\alpha_2} + p_{\alpha_2\beta_2}$ which corresponds to the sum of the transfer time of job β_2 from M_{α_2-1} to M_{α_2} and its processing time on M_{α_2}

Or,

(*ii*) $\alpha_1 = \alpha_2 + 1$ and $\beta_1 = \beta_2 - 1$, the weight of this arc is $c(u, v) = \tau_{\alpha_1,\alpha_1+1} + \tau_{\alpha_1+1,\alpha_1-1}$ which corresponds to the total duration of a loaded move from M_{α_1} to M_{α_1+1} and an empty move from M_{α_1+1} to M_{α_1-1}.

Fact 1 Let $\mathcal{P} = (u_1, ..., u_p)$ be a path in $\hat{G}$ starting at node $u_1 \equiv (\alpha_1, \beta_1)$ and ending at node $u_p \equiv (\alpha_p, \beta_p)$ and having a total weight $c(\mathcal{P})$. Then, $c(\mathcal{P})$ is a lower bound on the minimal time elapsed between the finishing time of $O_{\alpha_1\beta_1}$and the starting time of $O_{\alpha_p\beta_p}$.

Proof. This result follows from the fact that the weight of an arc $(u, v) \in \hat{A}$ (where $u \equiv (\alpha_1, \beta_1)$ and $w \equiv (\alpha_2, \beta_2)$) corresponds to the minimal time that elapses between the finishing time of operation $O_{\alpha_1\beta_1}$ and the starting time of operation $O_{\alpha_2\beta_2}$. ■

An immediate consequence is the following.

Fact 2 A valid lower bound on the total time that is required for completing all the jobs upon processing operation $O_{\alpha_1,\beta_1}$is equal to the sum of the value of the longest path in $\hat{G}$ between nodes $u_1 \equiv (\alpha_1, \beta_1)$ and $u^* \equiv (m, n)$ and the transfer time of the last scheduled job to the output device (i.e. $\tau_{m,m+1}$).

The main features of the branch-and-bound algorithm that we have implemented for finding an optimal sequence of the robot moves are the following.

- **Root node**: The root node N^0 corresponds to a partial schedule $s(N^0) = (O_{01}, O_{11})$ (obviously these two operations are necessarily scheduled first).

- **Branching strategy**: Given a node N having a corresponding partial schedule $s(N)$, a child node N^+ is created by selecting an unscheduled operation O_{ij} whose immediate predecessors $O_{i-1,j}$ and $O_{i+1,j-1}$ have been already included in the partial schedule $s(N)$. The partial schedule associated with N^+ is $s(N^+) = s(N)O_{ij}$.

- **Lower bound**: For each newly created node N, we compute the completion time of the last scheduled operation and then we use Fact 2 for deriving a lower bound $C(N)$ on the minimum completion time.

- **Upper bound**: An upper bound UB is computed at the root node by approximately solving the one-machine problem using a simple list scheduling algorithm: at each iteration schedule a candidate operation O_{ij} whose immediate predecessors have been already scheduled and whose starting time is minimal. Obviously, if for a node N we have $C(N) \geq UB$ then this node is pruned.

- **Search strategy**: We have implemented the following search strategy. We select amongst the candidate nodes the one corresponding to appending an unscheduled operation O_{ij} having the smallest origin machine index.

5 A genetic algorithm

During the last two decades, metaheuristics have become powerful tools for the approximate solution of intractable combinatorial optimization problems. In particular, genetic algorithms have been implemented for providing high-quality solutions to a wide variety of challenging flow shop problems (see for example Ruiz et al. (2006) and Rajkumar et al. (2009), to quote just a few). In this section, we investigate the ability of a genetic algorithm (hereafter, referred to by $GA1$) to effectively solve the robotic cell problem. Next, the performance of $GA1$ will be compared against the performance of the proposed approximate decomposition algorithm. The main features of $GA1$ are the following :

- **Solution encoding** : An ordered list (i.e. permutation) $\sigma = (\sigma(1), ..., \sigma(n))$ of the n jobs is used to represent a chromosome.

- **Fitness computation** : Given a chromosome $\sigma = (\sigma(1), ..., \sigma(n))$, the fitness of the corresponding solution is computed as follows. First, the m machines are sequenced according the ordering that is specified by the chromosome. Second, the sequence of the robot moves is obtained using the afore-mentioned list scheduling algorithm: at each iteration, we schedule the robot operation whose predecessors have been already scheduled and whose starting time is minimal. In so doing, a feasible schedule is obtained. Clearly, we might use a more sophisticated approach for scheduling the robot moves, but this would require longer CPU times.

- **The crossover operator** : We implemented a powerful crossover operator that was recently introduced by Ruiz et al. (2006) for solving the (standard) permutation flow shop problem. This operator is called *Similar Job Order Crossover* (or, SJOX for short) and is

described as follows. Given two parent chromosomes, a crossing point is randomly selected along the length of the first parent, and an offspring is produced by copying the identical jobs that are located at the same positions in both parents into the corresponding positions of it. Then missing jobs before the crossing point are inherited from the first parent. Lastly, the missing jobs are copied in the relative order of the second parent. Clearly, if no identical jobs are at the same position in the parents, the crossover operator will behave like the one-point crossover.

- **The mutation operator**: The performance of a GA might be significantly improved when it is hybridized with a local search procedure. In our GA, instead of using a classical mutation operator, we implemented the following interchange procedure. First, a chromosome is chosen randomly and is mutated with a specified probability. Then, we perform a pairwise interchange of jobs (the interchanged jobs are not necessarily adjacent). Obviously, if this interchange leads to a reduction of the makespan then the chromosome is replaced with the improved one. The process is continued until no improvement could be achieved.

- Parameter setting: We performed preliminary experiments and found that a good efficacy is achieved by setting the parameters as follows:

- Population size = 300 (The set of initial chromosomes are randomly generated)
- Crossover probability = 0.9
- Mutation probability = 0.4
- Maximum number of generations = 1000
- Maximum number of consecutive non improving generations before stopping = 100.

Remark 2 *In order to accelerate the convergence of the branch-and-bound algorithm that is described in Section 3.2, we implemented an additional genetic algorithm GA2 that delivers an upper bound for* $F|Block, t_k|C_{\max}$. *GA2 is very similar to GA1. The only difference lies in the fitness computation. In GA2, the fitness of a chromosome* $\sigma = (\sigma(1), ..., \sigma(n))$ *is set equal to the makespan of the corresponding permutation.*

6 Computational results

In order to assess the empirical performance of the proposed two-phase approach, we have coded all the proposed procedures and carried out computational experiments on a set of randomly generated *RCP* instances. The test-bed was generated in the following way. The processing times are drawn from the discrete uniform distribution on $[1, 100]$ and the transportation time between a pair of machines M_i and M_k is $\tau_{i,k} = 2 \times |i - k|$. The proposed algorithms were coded and compiled with Microsoft Visual Studio C^{++} (version 6.0). The

linear programming relaxation that is used for computing $LB4$ has been solved using CPLEX 9.1. All the computational experiments were carried out on a Pentium IV 3.2 GHz, 1 Gbytes RAM PC.

6.1 Performance of the lower bounds

In this section, we present an empirical analysis of the performance of the proposed lower bounds for the flow shop problem with blocking and transportation times that is solved in Phase 1.

The number of jobs n is taken equal to $n = 15$, 20, 25, 30, 40, and 50. The number of machines m is taken equal to 3, 4, and 5. For each (m, n) combination, 10 instances were randomly generated. In Table 1, we report for each lower bound LBi $(i = 1, ..., 5)$ and $MAX\text{-}LB \equiv \max_i\{LBi\}$ the average percentage deviation from the value UB_{GA2} of the solution of $GA2$ (i.e. $100 \times (UB_{GA2} - LBi)/LBi$).

Insert Table 1 here

We see from this table that $LB4$, $LB4^*$, and $LB5$ exhibit the smallest percentage deviation**.** However, we observe that the performance of all the lower bounding strategies deteriorates as the number of machines increases. Interestingly, we see that for $m = 5$, the one-machine based lower bounds outperform the two-machine based lower bound $LB5$.

In addition, we found that $LB1$, $LB2$, $LB3$, $LB4^*$, and $LB5$ are extremely fast since they require negligible CPU times (< 0.001 sec). On the contrary, $LB4$ which requires invoking an LP solver takes relatively longer CPU times.

In our experiments, we found that the equality $LB4^* = LB4$ holds for almost all instances (178 out of 180 instances). In Table 2, we report for each of these two bounds the mean CPU time in seconds.

Insert Table 2 here

We see from this table, that although $LB4$ and $LB4^*$ exhibit a very similar performance, the latter requires significantly shorter CPU times.

6.2 Performance of the two-phase approach

We have carried out extensive computational experiments in order to determine which lower bound provides the best performance after being embedded within the branch-and-bound algorithm of Phase 1. We found that a good trade-off is achieved by using the following hybrid strategy: if the level of the node is smaller than $\frac{2*n}{3}$ then $LB4^*$ is invoked, otherwise $LB5$ is used. Here again, for each combination (m, n) we randomly generated 10 instances.

Table 3 displays a summary of the computational results of the proposed two-phase method. For each combination (m, n), we provide:

- TNO : Total number of operations which is equal to $2nm + n$ (it includes nm machine operations and $n(m + 1)$ robot operations
- $US1$: number of instances for which optimality was not proved in Phase 1 after reaching a 2-hour time limit
- $Time\ 1$: mean CPU time of Phase 1
- $US2$: number of instances for which optimality was not proved in Phase 2 after reaching a 5-min time limit
- $Time\ 2$: mean CPU time of Phase 2
- $Time\ GA2$: mean CPU of $GA2$
- TT : mean total CPU time (viz. $TT = Time\ 1 + Time\ 2 + Time\ GA2$)
- *Gap* $GA2$: mean percentage deviation of the solution provided by $GA2$ with respect to the optimal makespan that is computed in Phase 1
- *Gap* $GA1$: mean percentage deviation of the solution provided by $GA1$ with respect to the optimal makespan that is computed in Phase 1
- *Gap* $2P$: mean percentage deviation of the solution provided by the two-phase method with respect to the optimal makespan that is computed in Phase 1**.**

Insert Table 3 here

Looking at Table 3, we see that the proposed approach exhibits a good overall performance since it consistently delivers high-quality solutions. Indeed, the average gap over the 210 instances is 0.85%. Moreover, we observe that for 96.7% of the instances, Phase 1 provided proven optimal solutions. However, the branch-and-bound algorithm in Phase 1 failed to deliver proven optimal solutions for one 4-machine and six 5-machine instances (out of 210 instances). Nevertheless, this is a remarkable result since it is well-documented that similar flow shop problems with blocking are notoriously hard to solve to optimality. For instance, Ronconi (2005) reported that her branch-and-bound algorithm failed to solve 9 out of 10 $F|Block|C_{\max}$ instances with 20 jobs and 4 machines. On the other hand, we found that the branch-and-bound algorithm of Phase 2 failed to deliver proven optimal solutions for 47 instances (out of 210) after reaching the 5-minute time limit. This provides a clear evidence of the hardness of the one-machine problem that is solved in Phase 2. Interestingly, we observe from Table 3 that the two-phase method consistently outperforms the genetic algorithm for 17 problem size combinations (out of 21). More precisely, we found that the two-phase method outperformed the genetic algorithm for 50% of the instances, and that it produced inferior solutions for 28% of the instances. However, the genetic algorithm provides

in most cases reasonably good solutions while requiring significantly shorter CPU times. Indeed, the largest instances often require about 1 minute CPU time only. Consequently, while the two-phase approach constitutes an appealing method for efficiently solving small and medium-size RCP instances, the genetic algorithm proves much more appropriate for addressing large-size instances.

7 Conclusion

In this paper, we have investigated the robotic cell problem. We have proposed an approximate decomposition algorithm. To the best of our knowledge, this is the first attempt to solve this complex scheduling problem. The proposed approach breaks the problem into two scheduling problems: a flow shop problem with blocking and transportation times and a single machine problem with precedence constraints, time lags, and setup times. We have proposed for each of these two problems an exact branch-and-bound algorithm. In order to assess the performance of the proposed approach, we have proposed a genetic algorithm that includes a powerful crossover operator as well as a local search-based mutation operator. We reported the results of a computational study that provide evidence that the proposed two-phase approach is robust, consistently delivers high-quality solutions, and outperforms the genetic algorithm. However, we found that the genetic algorithm requires significantly shorter CPU times and proves useful for efficiently solving large-scale instances.

Scheduling robotic cells constitute a challenging class of scheduling problems, and despite its pertinence to the fast growing topic of FMSs, its in-depth investigation is still in its infancy. We believe that the development of exact as well as tailored sophisticated metaheuristics for this important problem class is a promising avenue for future research.

Acknowledgments - The authors would like to express their gratitude for a number of careful remarks and comments by two anonymous referees that led to improvements to the paper. Jacques Carlier and Aziz Moukrim would like to thank the ANR for its financial support (LMCO project). Dr. Mohamed Haouari would like to thank Fatimah Alnijris Research Chair for Advanced Manufacturing Technology for the financial support provided for this research.

References

[1] Abadi, K., Hall, N. and Sriskandarajah, C. (2000). Minimizing cycle time in a blocking flowshop. *Operations Research* 48, 177-180.

[2] Agnetis, A. (2000). Scheduling no-wait robotic cells with two and three machines. *European Journal of Operational Research* 123, 303-314.

[3] Agnetis, A. and Pacciarelli, D. (2000). Part sequencing in three-machine no-wait robotic cells. *Operations Research Letters* 27, 185-192.

[4] Aneja,Y.P., and Kamoun, H. (1999). Scheduling of parts and robot activities in a two machine. *Computers & Operations Research* 26, 297-312.

[5] Blazewicz, J. Eiselt, H.A. Finke, G. Laporte, G. and Weglarz, J. (1991). Scheduling tasks and vehicles in a flexible manufacturing system. *International Journal of Flexible Manufacturing Systems* 4, 5-16.

[6] Caraffa, V. Ianes, S. Bagchi, T.P. and Sriskandarajah, C. (2001). Minimizing makespan in a blocking flowshop using genetic algorithms. *International Journal of Production Economics* 70, 101-115.

[7] Crama, Y. Kats, V. Van de Klundert, J. and Levner E. (2000). Cyclic scheduling in robotic flowshops. *Annals of Operations Research* 96, 97-124.

[8] Desrochers, M. and Laporte, G. (1991). Improvements and extensions to the Miller-Trucker-Zemlin subtour elimination constraints. *Operations Research Letters* 10, 27-36.

[9] Dawande, M.W., Geismar, H.N., Sethi, S.P., Sriskandarajah, C. (2007). Throughput Optimization in Robotic Cells, Springer.

[10] Dawande, M.W, Geismar, H.N., Sethi, S.P., and Sriskandarajah, C. (2005). Sequencing and scheduling in robotic cells: Recent developments. *Journal of Scheduling* 8, 387-426.

[11] Dutta, S.K., and Cunningham, A.A. (1975). Sequencing two-machine flowshops with finite intermediate storage. *Management Science* 21, 989-996.

[12] Ganesharajah, T., Hall, N.G., and Sriskandarajah, C. (1998). Design and operational issues in AGV-served manufacturing systems. *Annals of Operations Research* 76 (1), 109-154.

[13] Grabowski, J. and Pempera, J. (2007). The permutation flow shop problem with blocking, A tabu search approach. *Omega* 35, 302-311.

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

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

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

[17] Hurink, J. and Knust, S. (2002). A tabu search algorithm for scheduling a single robot in a job-shop environment. *Discrete Applied Mathematics* 119, 181-203.

[18] Hurink, J. and Knust, S. (2001). Makespan minimization for flow-shop problems with transportation times and a single robot. *Discrete Applied Mathematics* 112, 199-216.

[19] Knust, S. (1999). Shop-scheduling problems with transportation. PhD thesis, Fachbereich Mathematik/Informatik, University of Osnabrück.

[20] Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., and Shmoys, D. (1993). Sequencing and Scheduling: Algorithms and Complexity. *Handbooks in Operations Research and Management Science* 4, S.C. Graves, A.H.G. Rinnooy Kan, P. Zipkin (eds.), 445-522.

[21] Lee, C.-Y. and Chen, Z.-L. (2001). Machine scheduling with transportation considerations. *Journal of Scheduling* 4, 3-24.

[22] Levner, E.M. (1969). Optimal planning of parts machining on a number of machines. *Automation and Remote Control* 12, 1972-1978.

[23] Mascis, A. and Pacciarelli, D. (2002). Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research* 143, 498-517.

[24] Mc Cormick, S.T., Pinedo, M.L. Shenker, S. and Wolf, B. (1989). Sequencing in an assembly line with blocking to minimize cycle time. *Operations Research* 37, 925-935.

[25] Miller, C., Tucker, A., and Zemlin, R. (1960). Integer programming formulations and travelling salesman problems. *J. ACM* 7, 326-329.

[26] Pinedo, M.L. (2008). Scheduling theory, algorithms, and systems. Springer.

[27] Rajkumar, R., Shahabudeen, P. (2009). An improved genetic algorithm for the flowshop scheduling problem. *International Journal of Production Research* 47, 233–249.

[28] Ruiz, R., Maroto, C., and Alcaraz, J. (2006). Two new robust genetic algorithms for the flowshop scheduling problem. *Omega* 34, 461-476.

[29] Reddi, S.S. and Ramamoorthy, C.V. (1972). On the flowshop sequencing problems with no-wait in process. *Operational Research Quarterly* 23, 323-330.

[30] Ronconi, D.P. (2005). A branch-and-bound algorithm to minimize the makespan in a flowshop with blocking. *Annals of Operations Research* 138, 53-65.

[31] Ronconi, D.P. and Armentano V.A. (2001). Lower bounding schemes for flowshops with blocking in-process. *Journal of the Operational Research Society* 52, 1289-1297.

[32] Roy, B. and Sussman, B. (1964). Les problèmes d'ordonnancement avec contraintes disjonctives. Note DS No. 9 bis, SEMA, Paris.

[33] Sriskandarajah, C., Hall, N.G., and Kamoun, H. (1998). Scheduling large robotic cells without buffers. *Annals of Operations Research* 76, 287-321.

[34] Vairaktarakis, G.L. (2003). Simple algorithms for Gilmore-Gomory's traveling salesman and related problems. *Journal of Scheduling* 6, 499-520.