RAIRO-Oper. Res. 51 (2017) 805–818
DOI: 10.1051/ro/2016064

RAIRO Operations Research
www.rairo-ro.org

THROUGHPUT OPTIMIZATION FOR THE ROBOTIC CELL PROBLEM WITH CONTROLLABLE PROCESSING TIMES

Mohammed Al-Salem[1] and Mohamed Kharbeche[2]

Abstract. In this paper, we present a MIP-based heuristic and an effective genetic algorithm for the Robotic Cell Problem with Controllable Processing Times (RCPCPT). This problem arises in modern automated manufacturing systems and requires simultaneously scheduling jobs, machines, and transportation devices in order to maximize the throughput or minimize the makespan. The RCPCPT is modeled as a flow shop problem with blocking constraints, a single transport robot, and controllable processing times. This latter feature of the model refers to the fact that the processing times are not fixed but vary linearly with the acceleration cost and therefore should be determined as part of the problem output. We formulate the problem as a nonlinear mixed-integer programming formulation and we use its linearized form to derive LP- and MIP-based heuristics. In addition, we proposed a genetic algorithm consistently yields near-optimal solution and it encompasses several novel features including, an original solution encoding as well as a mutation operator that requires iteratively solving MIPs in order to generate feasible processing times. Finally, we present a computational study for the proposed formulation, heuristics and genetic algorithm and we provide an empirical evidence of the effectiveness of the MIP-based heuristic for small instances and the genetic algorithm for large instances.

Mathematics Subject Classification. 49-XX.

Received June 1, 2016. Accepted August 14, 2016.

1. Introduction

Since the publication by Johnson [18] of his seminal paper on the two-machine flow shop problem, literally thousands of papers dealing with various scheduling models have been published in the scientific literature. However, a glaring fact in this context is that a great majority of these papers address models where the processing times are assumed to be constant parameters, and part of the problem input. Nevertheless, in many real-life situations, the processing times may be controllable (that is, increased or decreased) by allocating resources (energy, workforce, money). In such situations, the processing times are considered as *decision variables* and therefore should be determined as part of the solution output.

In this paper, we investigate a variant of the flow shop problem with controllable processing times that arises in flexible manufacturing systems. More precisely, we address the *Robotic Cell Problem with Controllable Processing Times* (RCPCPT) that is defined as follows and illustrated by Figure 1. We are given a set J of n

Keywords. Robotic cell, flow shop, controllable processing times, MIP-base heuristic, genetic algorithm.

[1] Department of Mechanical and Industrial Engineering, College of Engineering, Qatar University, Doha, Qatar.
alsalem@qu.edu.qa

[2] Qatar Transportation and Traffic Safety Center, Qatar University, Doha, Qatar.
mkharbec@qu.edu.qa

Article published by EDP Sciences

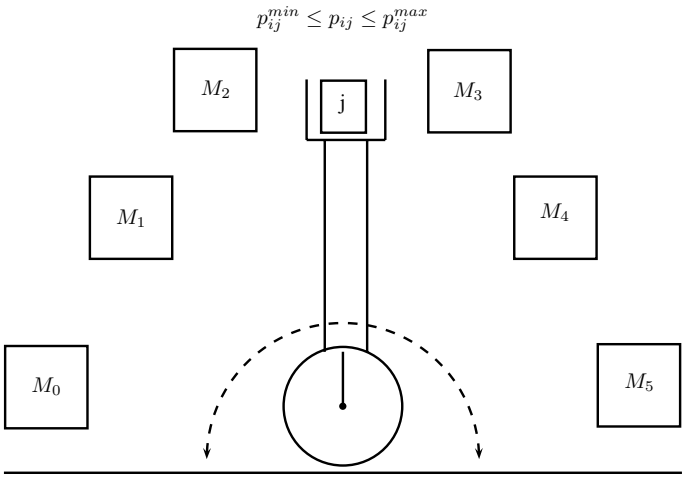

FIGURE 1. Robotic Cell with 4 machines.

jobs where each job has to be processed in a robotic cell. The robotic cell consists of a set of m machines M_1, M_2, ..., M_m, an input buffer M_0, an output buffer M_{m+1}, and a single robot that is used for transferring the jobs between different machines. At time $t = 0$, all jobs are available at the input device M_0. Each job $j \in J$ has to be processed nonpreemptively on machines M_1, M_2, ..., M_m, in that order, and then transferred to the output device M_{m+1}. The robot can transfer at most one job at any time and the duration of a robot move from M_i to M_h $(i, h = 0, \ldots, m+1)$ is deterministic and requires τ_{ih} units of time. The machines have neither input nor output buffering facilities. Consequently, after processing a job j on machine M_i $(i = 1, \ldots, m)$, this latter remains blocked until the robot picks-up j and transfers it to the subsequent 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}). It is noteworthy that, because of the blocking constraint, passing is not possible and therefore only permutation schedules are considered. Furthermore, at any time each machine can process at most one job and each job can be processed on at most one machine. An important feature of the investigated problem is that the processing time p_{ij} of operation O_{ij} of job $j \in J$ on machine M_i $(i = 1, \ldots, m)$ is not fixed *a priori* but is assumed to be a decreasing linear function of the *acceleration cost* c_{ij} that is allocated to the processing of O_{ij}. More precisely, the data that is associated with each operation O_{ij} includes three parameters: (i) a non-compressed (maximum) processing time $p_{ij}^{\max}$, (ii) a compressed (minimum) processing time $p_{ij}^{\min}$, and (iii) a compression rate a_{ij}. Hence, the processing times are given by:

$$p_{ij} = p_{ij}^{\max} - a_{ij}c_{ij}, \qquad \forall j \in J \text{ and } i = 1, \ldots, m, \tag{1.1}$$

where the acceleration cost c_{ij} satisfies:

$$0 \leq c_{ij} \leq c_{ij}^{\max} \equiv (p_{ij}^{\max} - p_{ij}^{\min})/a_{ij}, \qquad \forall j \in J \text{ and } i = 1, \ldots, m. \tag{1.2}$$

Furthermore, we assume that the total acceleration cost should not exceed a preset budget B. It is noteworthy that this linear resource consumption function is very popular in the scheduling literature dealing with controllable processing times (see Janiak [17], Zdrzalka [43], and Biskup and Cheng [5], to quote just a few pioneering contributions).

Hence, the RCPCPT requires to simultaneously determine:

(i) the processing time of each operation O_{ij} $(j \in J, i = 1, \ldots, m)$;
(ii) the processing order of the n jobs; and

(iii) the sequence of robot moves (including both empty as well as loaded moves).

The objective is to minimize the time $C_{\max}$ at which all the jobs are completed (makespan). This objective is equivalent to maximizing the productivity of the manufacturing system or throughput.

The RCPCPT is considered as a generalization of the classic robotic cell without controllable processing times and is strongly $\mathcal{NP}$-hard. In particular, Hall and Sriskandarajah [15] prove that the robotic cell problem is strongly $\mathcal{NP}$-hard for $m \geq 3$.

The robotic cell problem with controllable processing times arises in Flexible Manufacturing Systems (FMSs), which are highly automated production systems capable of producing a wide variety of job types. Our motivation for the study of this complex problem stems from the fact that one of the most crucial 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. During the last decades, the rapid development of robotic cells in various manufacturing industrial setting has prompted the investigation of an ever-growing number of new scheduling problems. We refer to the comprehensive book of Dawande *et al.* [7] for a review of sequencing and scheduling problems arising in robotic cells. However, at this point it is worth emphasizing that the great majority of previously investigated robotic cell scheduling problems deal with *cyclic scheduling problems* with constant processing times and machines producing a family of similar parts, in a steady-state. Nevertheless, few recent papers addressed a non-cyclic multiple-part-type robotic cell problem (with constant processing times). Park [27] studied the scheduling of a robot in a flexible manufacturing cell problem where parts are arriving randomly. He presented a simulation-based analysis in order to determine the best movement decision for the robot. Carlier *et al.* [6] proposed an approximate decomposition algorithm to the robotic cell problem. The proposed approach decomposes the problem into two scheduling problems: a flow shop problem with blocking and transportation times and a single machine problem (that corresponds to the robot sequencing) with precedence constraints, time lags, and setup times. Each of these two problems is solved using an exact branch-and-bound algorithm. Furthermore, Kharbeche *et al.* [21] proposed an exact branch-and-bound algorithm approach for the same problem and found that instances with up to 16 jobs and 5 machines can be optimally solved.

To the best of our knowledge, the literature on scheduling flexible cells with controllable processing times is relatively scant. Al-Salem *et al.* [4] studied the same problem and presented a free-slack-based genetic algorithm for the case of linear resource consumption function. The authors showed that the proposed algorithm provides good solution. Gultekin *et al.* [13,14] and Yilidiz *et al.* [42] addressed a cyclic problem of scheduling two identical CNC machines and a material handling robot where two objectives are considered: minimizing the cycle time and the total manufacturing cost. Akturk and Ilhan [1] addressed a CNC machine scheduling with controllable processing times where the objective is to minimize the sum of total weighted tardiness, tooling and machining costs. They formulated the problem as nonlinear mixed-integer program and solved it using a heuristic approach. Uruk *et al.* [33] investigated a bi-objective two-machine flow shop scheduling problem with flexible operations. A heuristic procedure is proposed to solve practical-sized instances.

By contrast, we observe that during the last decade a large number of researchers have been investigating "more standard" scheduling problems with controllable processing times. At this point, it is worth mentioning that, in addition to the aforementioned linear job processing times model (1.2), some authors (see Koulamas *et al.* [22], Shabtay and Steiner [29]) have considered an alternative convex resource consumption function that is given by

$$p_{ij} = \left(\frac{w_{ij}}{c_{ij}}\right)^{\alpha}, \qquad \forall j \in J \text{ and } i = 1, \ldots, m, \tag{1.3}$$

where w_{ij} is a positive parameter that represents the workload of operation O_{ij} and α is a positive parameter.

Actually, most of the papers so far published deal with one-machine scheduling problems, and to a lesser extent with parallel machines, flow shop, and job shop problems. We refer to Shabtay and Steiner [29] for a comprehensive review of these scheduling models. In addition to the numerous references that are quoted in this latter review paper, some further contributions in this area have been recently published with a noticeable

dominance of papers dealing with various single-machine scheduling problem with controllable processing times (see Koulamas *et al.* [22], Xu *et al.* [36], Karimi-Nasab and Ghomi [19], Yin *et al.* [40,41], Yin and Wang [39], Xu *et al.* [35,37], Yang *et al.* [38]). In addition, parallel machine problems with controllable processing times received attention (see Li *et al.* [23], Mor and Mosheiov [24], Shioura *et al.* [32], Kayvanfar *et al.* [20]). Furthermore, flow shop and job shop scheduling problem with controllable processing times were recently investigated by Niu *et al.* [26], Renna [28] and Shabtay *et al.* [30].

Finally, genetic algorithms have remarkable ability in solving scheduling problems and many similar hard optimization problems. Recently, they have demonstrated their ability in solving various problems that arises from the maritime industry [2,3,9,16,44] and logistics optimization problems [8,10–12].

In this paper, we make the following contributions:

- We present a nonlinear programming formulation of the RCPCPT. Next, this formulation is linearized and an equivalent mixed-integer linear programming model is derived. This model can be useful for optimally solving small-sized instances or for computing lower bounds.
- We demonstrate the practical usefulness of the proposed model by using it to derive LP- and MIP-based heuristics. These heuristics can be used (at little coding effort) for heuristically solving large RCPCPT.
- We propose a MIP-based genetic algorithm that encompasses several innovative features including an original solution encoding as well as a mutation operator that requires iteratively solving restrictions of the proposed MIP formulation. This approach can be used to derive high-quality solutions for large problems.
- We present a computational study of the proposed heuristics. This study provides an empirical evidence of the effectiveness of the proposed solution procedures.

The remainder of this paper is organized as follows. In Section 2, we present a mathematical programming formulation for the RCPCPT. In Section 3, we derive a LP- as well as MIP-based heuristics. In Section 4, we provide a detailed description of the components of a genetic algorithm. The results of a comprehensive study are described and analyzed in Section 5. Finally, Section 6 provides a summary of the paper along with some concluding remarks.

2. Formulation of the RCPCPT

In this section, we describe a valid nonlinear formulation for the RCPCPT. Then, we proposed the linearized form of the problem.

2.1. A mixed-nonlinear integer programming formulation

A similar formulation (but linear) was previously proposed for the robotic cell problem with fixed processing times (Kharbeche *et al.* [21]). To begin with, we present the decision variables and the nonlinear mixed-integer programming formulation. Next, we linearize the developed model by incorporating additional variables.

We define the set Π, as the set of the robot loaded moves. A robot operation that corresponds to a transfer of the job that is scheduled on the kth position from M_i to M_{i+1} is denoted by θ_{ik} $(i = 0, \ldots, m,\ k = 1, \ldots, n)$. Clearly, robot moves operations are interrelated by precedence relationships. More precisely, because of the blocking constraints, θ_{ik} should be preceded by operation $\theta_{i+1,k-1}$ $(i = 0, \ldots, m,\ k = 2, \ldots, n)$. Also, as a consequence of the flow shop constraint, θ_{ik} should be preceded by operation $\theta_{i-1,k}$ $(i = 1, \ldots, m,\ k = 1, \ldots, n)$. For each $\theta_{ik} \in \Pi$, we denote by $pred(\theta_{ik})$ and $succ(\theta_{ik})$ the sets of predecessors and successors of θ_{ik}, respectively.

Decision variables:

x_{kj} : binary variable that takes value 1 if job j is assigned to the kth position in the schedule, and 0 otherwise, $\forall k = 1, \ldots, n,\ j \in J$;

y_{ik}^{lh} : binary variable that takes value 1 if the robot loaded move θ_{ik} is performed before the robot move θ_{lh}, and 0 otherwise, $\forall \theta_{ik}, \theta_{lh} \in \Pi$;

t_{ik} : starting time of operation θ_{ik}, $\forall i = 0, \ldots, m,\ k = 1, \ldots, n$;

p_{ij} : processing time of job j on machine M_i, $\forall i = 1, \ldots, m,\ j \in J$.

Model formulation:
Using these definitions, the RCPCPT can be stated as follows:

$$\text{(MINLP): Minimize } t_{mn} + \tau_{m,m+1} \tag{2.1}$$

Subject to:

$$\sum_{k=1}^{n} x_{kj} = 1, \qquad \forall j = 1, \ldots, n, \tag{2.2}$$

$$\sum_{j=1}^{n} x_{kj} = 1, \qquad \forall k = 1, \ldots, n, \tag{2.3}$$

$$t_{i+1,k} \geq t_{ik} + \tau_{i,i+1} + \sum_{j=1}^{n} p_{i+1,j} x_{kj}, \ \forall k = 1, \ldots, n, \ i = 0, \ldots, m-1, \tag{2.4}$$

$$t_{i-1,k} \geq t_{i,k-1} + \tau_{i,i+1} + \tau_{i+1,i-1}, \quad \forall k = 2, \ldots, n, \quad i = 1, \ldots, m, \tag{2.5}$$

$$\sum_{\theta_{lh} \in pred(\theta_{ik})} y_{lh}^{ik} = 1, \qquad \forall \theta_{ik} \in \Pi \setminus \{\theta_{01}\}, \tag{2.6}$$

$$\sum_{\theta_{lh} \in succ(\theta_{ik})} y_{ik}^{lh} = 1, \qquad \forall \theta_{ik} \in \Pi \setminus \{\theta_{mn}\}, \tag{2.7}$$

$$t_{i-1,k} \geq t_{i+1,k-1} + \tau_{i+1,i+2} + \tau_{i+2,i-1} + M(y_{i+1,k-1}^{i-1,k} - 1), \tag{2.8}$$
$$\forall k = 2, \ldots, n, \ i = 1, \ldots, m-1,$$

$$t_{i+1,k-1} \geq t_{i-1,k} + \tau_{i-1,i} + \tau_{i,i+1} + M(y_{i-1,k}^{i+1,k-1} - 1), \tag{2.9}$$
$$\forall k = 2, \ldots, n, \ i = 1, \ldots, m-1,$$

$$t_{0,k} \geq t_{m,k-p} + \tau_{m,m+1} + \tau_{m+1,0} + M(y_{m,k-p}^{0,k} - 1), \tag{2.10}$$
$$\forall k = 2, \ldots, n, \ p = 1, \ldots, m-1,$$

$$t_{0,k+p} \geq t_{m,k-1} + \tau_{m,m+1} + \tau_{m+1,0} + M(y_{m,k-1}^{0,k+p} - 1), \tag{2.11}$$
$$\forall k = 2, \ldots, n, \ p = 1, \ldots, m-1,$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \frac{p_{ij}}{a_{ij}} \geq \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{p_{ij}^{\max}}{a_{ij}} - B, \tag{2.12}$$

$$p_{ij}^{\min} \leq p_{ij} \leq p_{ij}^{\max}, \qquad \forall i = 1, \ldots, m, \ j \in J, \tag{2.13}$$

$$t_{ik} \geq 0, \ \forall i = 0, \ldots, m, \ k = 1, \ldots, n, \tag{2.14}$$

$$x_{kj} \in \{0, 1\}, \qquad \forall k = 1, \ldots, n, \quad j \in J, \tag{2.15}$$

$$y_{ik}^{lh} \in \{0, 1\}, \qquad \forall \theta_{ik}, \theta_{lh} \in \Pi. \tag{2.16}$$

The objective (2.1) is to minimize the makespan. Constraints (2.2) and (2.3) require that each job is assigned to exactly one position in the schedule, and each position is assigned to exactly one job, respectively. Constraint (2.4) is the flow shop constraint: it requires that $\theta_{i+1,k}$ is scheduled after achieving θ_{ik} and processing operation O_{ik} on M_i. Constraint (2.5) is the blocking constraint: $\theta_{i-1,k}$ is scheduled after $\theta_{i,k-1}$. Constraint (2.6) enforces that each robot operation (but, θ_{01}) has exactly one successor. Similarly, Constraint (2.7) requires that each robot operation (but, θ_{mn}) has exactly one predecessor. Constraints (2.8)–(2.11) (where M is a large integer) ensure that the precedence constraints between particular robot operations must be satisfied. Constraint (2.12) requires that the total acceleration costs should not exceed the preset budget and (2.13) sets

the upper and lower bounds for the processing times. Finally, (2.14)–(2.16) impose that the time variables are continuous and the x- and y-variables are binary.

2.2. Reformulation and linearization

Because of the nonlinearity of (2.4), Model (2.1)–(2.16) is a mixed-integer nonlinear program (MINLP). However, (MINLP) are widely known to be extremely difficult to solve. Furthermore, relying on a nonlinear package (such as BARON or SNOPT) for computing lower bounds, especially since many nonlinear programs are non-robust, could falsely inflate lower bounds. An alternative strategy for solving Model (2.1)–(2.16) requires deriving an equivalent mixed-integer *linear* programming formulation in the spirit of the Reformulation Linearization Technique (RLT) of Sherali and Adams [31]. To that aim, we introduce an additional decision variable π_{ijk} that is defined as follows:

$$\pi_{ijk} = p_{ij}x_{kj}, \qquad \forall i = 1, \ldots, m, \quad j, k = 1, \ldots, n. \tag{2.17}$$

Using (2.2), we construct the following equality:

$$\left[\sum_{k=1}^{n} x_{kj} = 1\right] \times \; p_{ij}. \tag{2.18}$$

Also, using the bounding constraint (2.13), we construct the following inequalities:

$$\left[p_{ij}^{\min} \leq p_{ij} \leq p_{ij}^{\max}\right] \times \; x_{kj}.$$

The foregoing reformulation-linearization process yields the following equivalent MILP formulation of the RCPCPT:

$$\text{(MILP): Minimize } t_{mn} + \tau_{m,m+1} \tag{2.19}$$

subject to: (2.2)–(2.3), (2.5)–(2.12), (2.14)–(2.16), and

$$t_{i+1,k} \geq t_{ik} + \tau_{i,i+1} + \sum_{j=1}^{n} \pi_{i+1,jk}, \qquad \forall k = 1, \ldots, n, \quad i = 0, \ldots, m-1, \tag{2.20}$$

$$\sum_{k=1}^{n} \pi_{ijk} = p_{ij}, \quad \forall i = 1, \ldots, m, \quad j \in J, \tag{2.21}$$

$$p_{ij}^{\min} x_{kj} \leq \pi_{ijk} \leq p_{ij}^{\max} x_{kj} \quad \forall i = 1, \ldots, m, \; j \in J, \; k = 1, \ldots, n, \tag{2.22}$$

$$\pi_{ijk} \geq 0, \; , \quad \forall i = 1, \ldots, m, \; j \in J, \; k = 1, \ldots, n. \tag{2.23}$$

For convenience, in the sequel, we shall refer to a job that is scheduled at position k of the schedule as the kth job.

3. MIP-based heuristic approaches

The RCPCPT is a hard scheduling problem and it is therefore hardly conceivable that the proposed formulation can be used for optimally solving large-scale instances. Nevertheless, in this section we show that it can be advantageously used to derive approximate solutions at a little coding and time efforts.

3.1. A LP-based heuristic

First, we describe the following LP-based heuristic:

Step 1 – Solution of the LP relaxation: solve the LP relaxation of Model (MILP). Let $(\bar{x}, \bar{y}, \bar{t},\ \bar{p}, \bar{\pi})$ denote the optimal continuous solution.
Step 2 – Computation of the approximate job positions: set $\lambda_j = \sum_{k=1}^{n} k\bar{x}_{kj}$ for $j \in J$.
Step 3 – Construction of a job sequence: let $\sigma = (\sigma(1), \ldots, \sigma(n))$ denote the job permutation that is obtained by ordering the jobs according non-decreasing λ_j (ties are broken arbitrarily).
Step 4 – Scheduling of the robot moves: given the processing times vector $\bar{p}$ and the job permutation σ, determine a sequence of robot moves by solving the following MIP:

$$(RM)\ \text{Minimize } t_{mn} + \tau_{m,m+1} \tag{3.1}$$

Subject to:

$$t_{i+1,k} \geq t_{ik} + \tau_{i,i+1} + \bar{p}_{i+1,\sigma(k)}, \quad \forall k = 1, \ldots, n, \quad i = 0, \ldots, m-1, \tag{3.2}$$

$$(2.5)\text{-}(2.11),\ (2.14),\ (2.16). \tag{3.3}$$

Hence, the solution of the LP relaxation of Model (MILP) is not only used to compute the processing times but also provides an important information about the jobs sequencing. Indeed, the λ_j's that are computed in Step 2 are interpreted as approximate estimates of the job positions in the schedule. This sequence is generated, in Step 3, through ranking the jobs in non-decreasing λ_j. In the final step, we determine an approximate schedule of the robot moves by optimally solving Model (RM). It is noteworthy that we found that this latter problem can be very quickly solved using a commercial MIP solver.

3.2. A MIP-based heuristic

This approach decomposes the solution process in two main steps:
Step 1 – Solution of a MIP relaxation: relax the integrality constraint of the y-variables and solve the corresponding MIP relaxation.
Step 2 – Scheduling of the robot moves: given the processing times and the job assignment that were obtained in Step 1 construct an optimal sequence of the robot moves by solving Model (RM).

Hence, in Step 1 we compute the job processing times as well as the job positions by solving a relaxation of the genuine problem, and in Step 2 we compute the optimal sequencing of the robot. It is noteworthy that, for the sake of computational efficacy, a nonzero optimality tolerance might be set for the solution the MIP relaxation.

4. A genetic algorithm

Interestingly, the great majority of the GAs that have been implemented for solving scheduling flow shop problems require finding a job permutation. In this case, a natural representation of a chromosome is an ordered list (*i.e.* permutation) of the n jobs, and the fitness is often straightforwardly computed. However, this representation is seemingly not appropriate for the RCPCPT. Indeed, given a permutation of n jobs, it is unclear how the complete solution (including the processing times) could be derived. In the sequel, we describe a novel GA for solving the RCPCPT. We successively describe the main features of our algorithm: the solution encoding, the procedure for generating an initial solution, the fitness computation, crossover operator and the mutation operator.

4.1. Solution encoding

An $m \times n$ matrix $P = (p_{ij})$ is used to represent a chromosome. In this matrix, entry p_{ij} represents the processing time of job j on M_i.

4.2. Generation of an initial population

First, we make the following observation:

Fact 1. There exist an optimal solution whose total acceleration cost is equal to the budget B.

Proof. It suffices to observe that if the total acceleration cost of an optimal schedule is strictly smaller than B, then we can crash some operations at an additional cost without elongating the makespan. □

Thus, we focus on generating initial chromosomes whose total acceleration cost is equal to (or slightly smaller than) the budget B. To that aim, we implemented the following algorithm:

Algorithm 1. *Generation of a feasible initial chromosome.*

Random generation of the initial acceleration costs
for $i = 1, \ldots, m$ **do**
  for $j = 1, \ldots, n$ **do**
    Randomly draw c_{ij} from $U[0, c_{ij}^{\max}]$
  end for
end for
Update of the acceleration costs
Set $C = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}$
for $i = 1, \ldots, m$ **do**
  for $j = 1, \ldots, n$ **do**
    Set $c_{ij} = \min(\frac{B}{C} c_{ij}, c_{ij}^{\max})$
  end for
end for
Computation of the processing times
for $i = 1, \ldots, m$ **do**
  for $j = 1, \ldots, n$ **do**
    Set $p_{ij} = p_{ij}^{\max} - a_{ij} c_{ij}$
  end for
end for

To generate an initial population of size S, this procedure is reiterated S times.

4.3. Fitness computation

The fitness of a chromosome P is equal to the makespan $C_{\max}(P)$ of the robotic cell problem instance that is obtained by fixing the processing times. For the sake of efficiency, an approximate value of $C_{\max}(P)$ is obtained through using a modified NEH heuristic (hereafter, referred to as MNEH). It is noteworthy that the celebrated NEH algorithm is considered as one of the most effective constructive heuristics for the permutation flow shop problem (see Vasiljevic and Danilovic [34]). Heuristic NEH includes the following steps:

To solve the RCPCPT, we modify the NEH procedure by invoking in Steps 2 and 3.1 a procedure for computing the makespan of a specific subsequence. Recall that if the job sequence is specified, then computing the makespan requires scheduling the robot moves. This latter problem, that is equivalent to a one-machine problem, is solved using a simple list scheduling algorithm: at each step, schedule a ready operation whose starting time is minimal.

Algorithm 2. *NEH heuristic* (*Nawaz et al.* [25]).

1: **Step 1**: Sort jobs by decreasing cumulative processing times $T_j = \sum_{i=1}^{m} p_{ij}$. Let $(j_1, j_2, \ldots, j_n)$ be the resulting ordered list.
2: **Step 2**: Consider the two subsequences (j_1, j_2) and (j_2, j_1). Denote by L the one having the shortest makespan. Set $s = 3$.
3: **Step 3**:
4: **for** $q = 3, \ldots, n$ **do**
5: **Step 3.1** Consider the s subsequences that are obtained by inserting job j_q into the s possible positions of the subsequence L. Denote by $L^{'}$ the one having the shortest makespan.
6: **Step 3.2** Set $L = L'$, $s = s + 1$.
7: **end for**
8: **Step 4**: Output the n-job sequence L.

4.4. Crossover operator

Crossover operator selects two chromosomes to produce two offsprings. Typically, crossover takes two parents, cuts their chromosome strings at a randomly chosen position, swaps the head (or tail) segments to produce two offspring. Obviously, this type of crossover is not compatible with our chromosome representation. Instead, we propose the following crossover operator. Given two feasible chromosomes P^s and P^h together with the corresponding acceleration costs (c_{ij}^s) and (c_{ij}^h), respectively. Randomly draw δ from $[0, 1]$. We obtain two new chromosomes P^α and P^β by setting $c_{ij}^\alpha = \delta c_{ij}^s + (1 - \delta)c_{ij}^h$ and $c_{ij}^\beta = (1 - \delta)c_{ij}^s + \delta c_{ij}^h$ and by using (1.2) for computing the corresponding processing times.

Remark 4.1. It is easy to check that since the offsprings P^α and P^β are convex combinations of P^s and P^h then they are feasible as well (that is, $\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^\alpha \le B$ and $\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^\beta \le B$).

4.5. The mutation operator

The proposed mutation operator is an iterative improvement procedure. It considers as an input a chromosome P and output a chromosome P' such that $C_{\max}(P') \le C_{\max}(P)$. To describe a basic improvement step, we consider a chromosome P having a corresponding schedule. It is easily realized that some noncritical activities in P could be elongated (and therefore the total acceleration cost is reduced) without increasing the makespan. On the other hand, the cost savings could be allocated to accelerate some critical operations and therefore (possibly) reducing the makespan. In so doing, we obtain a new set of feasible processing times and we define P' as the corresponding chromosome. A key observation is that this improvement step can be achieved by solving a MIP formulation (MO) that is similar to Model (MILP) but with one important difference: the job assignment is specified by the permutation that is associated to chromosome P. Thus, the computation of the new processing times as well as the associated optimal robot moves is achieved through solving the following model:

$$\text{MO: Minimize } t_{mn} + \tau_{m,m+1} \tag{4.1}$$

subject to:

$$(2.5)\text{–}(2.14),\ (2.16) \tag{4.2}$$

$$t_{i+1,k} \ge t_{ik} + \tau_{i,i+1} + p_{i+1,\sigma(k)}, \qquad \forall k = 1, \ldots, n, \quad i = 0, \ldots, m-1, \tag{4.3}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{p_{ij}}{a_{ij}} \ge \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{p_{ij}^{\max}}{a_{ij}} - B, \tag{4.4}$$

$$p_{ij}^{\min} \le p_{ij} \le p_{ij}^{\max}, \qquad \forall i = 1, \ldots, m,\ j \in J, \tag{4.5}$$

where $\sigma(k)$ represents the the k^{th} job in chromosome P.

After computing the new processing times and the robot moves, procedure MNEH is invoked to compute $C_{\max}(P')$. If $C_{\max}(P') < C_{\max}(P)$, then a new similar improvement step is achieved by elongating non-critical operations of P' and (possibly) accelerating some critical operations. This improvement process is reiterated until the procedure fails to produce a new chromosome with an improved makespan. In this case, the latter generated chromosome replaces P in the current population.

4.6. Parameters of the genetic algorithm

In our genetic algorithm, we tested different parameters and we found that solutions of higher quality are achieved using the following settings and control schemes:

- Population size = 30.
- Crossover probability = 0.7.
- Mutation probability = 0.1.
- Maximum number of generations = 30.
- Maximum number of consecutive non improving generations before stopping = 10.

In our implementation, GA stops either when a maximum number of 30 generations has been surpassed or when the best solution of the population has not been improved on over 10 consecutive generations.

5. Computational results

To evaluate the proposed solution approaches, we have coded them in Microsoft Visual C++ (2010). Also, we used the commercial solver CPLEX 12.6 to solve the proposed MIP formulations. All our experiments were run on a Pentium IV 2.4 GHz PC.

The test-bed we have used consists of randomly generated RCPCPT instances. The number of machines m considered are 3, 4 and 5. For each size, the number of jobs n is taken equal to 10, 15, 20, 30, 40 and 50 jobs. For each combination $(m \times n)$, 10 instances are generated. The transportation time between a pair of machines M_i and M_k is $2\,|i-k|$ $(i, k = 0, \ldots, m+1)$. The compression rates and the acceleration cost are integers chosen randomly from the interval $[1, 10]$, $[1, 5]$, respectively. The controllable processing times are generated as follows. For each operation O_{ij} $(i = 1, \ldots, m,\ j \in J)$, the non-compressed processing time $p_{ij}^{\max}$ are drawn from the discrete uniform distributions on $[60, 100]$ and the compressed processing time $p_{ij}^{\min} = p_{ij}^{\max} - a_{ij}c_{ij}$, respectively. Finally, for each instance, the budget was set to $B = \left\lceil 0.5 \sum_{i=1}^{m} \sum_{j \in J} c_{ij} \right\rceil$ (recall that, $c_{ij}^{\max} = (p_{ij}^{\max} - p_{ij}^{\min})/a_{ij}$, for $i = 1, \ldots, m,\ j \in J$).

We conducted two sets of experiments. In the first set, we assess the performance of the proposed MIP formulation, and in the second one we analyze the performance of the proposed heuristics.

First, we investigate the performance of Model (MILP). We set the maximum CPU time limit to 1800 s. The results are displayed in Table 1. In this table, each row corresponds to the average performance that was computed for 10 randomly generated instances. The column headings are as follows: m: number of machines, n: number of jobs, $\#B$: number of binary variables, $\#C$: number of continuous variables, $\#Cons$: number of constraints, $Time$: Total CPU time, Opt: percentage of solved instances, GAP: mean percentage gap of unsolved instances (if any).

We see from Table 1 that all 10-job instances were optimally solved while requiring short CPU times. On the other hand, only 30% of the 15-job and 5-machine instances were solved within the maximum CPU time. However, we observe the mean gap of unsolved instances is small.

Now, we turn our attention to analyzing the performance of the heuristics against the best known lower bound. For each heuristic H, and each instance I, let $C_{\max}^{H}(I)$ and $\bar{C}_{\max}(I)$ denote the makespan derived by H and the best makespan obtained over all heuristics, respectively, and $LB(I)$ denote a lower bound on the optimal

TABLE 1. Performance of the MIP formulation.

m	n	$\#B$	$\#C$	$\#Cons$	$Time$	Opt	GAP
3	10	221	357	836	3.32	100	0.00
	15	416	778	1732	654.45	100	0.00
4	10	340	477	1185	6.44	100	0.00
	15	605	1033	2416	562.43	70	0.18
5	10	519	597	1594	10.16	100	0.00
	15	894	1288	3200	1077.10	30	1.08

TABLE 2. Relative deviations and CPU times of the proposed heuristics.

		LP-based		*MIP-based*		*GA*	
m	n	GAP_{LB}	$Time$	GAP_{LB}	$Time$	GAP_{LB}	$Time$
3	10	7.37	0.11	0.57	0.64	3.33	2.37
	15	11.13	0.12	1.48	2.40	5.49	5.15
	20	13.10	0.15	3.92	4.67	7.02	7.94
	30	15.47	0.16	6.36	9.80	7.77	20.77
	40	17.25	0.19	7.10	56.88	8.53	25.07
	50	17.09	0.24	7.37	131.55	8.29	31.55
4	10	8.72	0.13	0.66	0.67	3.96	6.57
	15	10.62	0.15	1.24	3.39	5.66	13.27
	20	14.62	0.19	5.85	5.70	8.24	22.94
	30	17.18	0.23	7.81	20.37	9.16	28.65
	40	19.18	0.22	12.92	56.77	10.45	39.55
	50	18.87	0.24	11.07	141.29	10.05	39.75
5	10	8.30	0.18	0.58	0.87	4.70	10.07
	15	12.07	0.30	2.74	3.11	5.37	22.29
	20	16.58	0.26	7.78	5.50	9.33	28.66
	30	19.19	0.48	12.41	29.67	11.39	32.06
	40	19.88	0.30	16.71	104.61	11.99	71.49
	50	20.14	0.38	15.14	127.51	11.55	103.97
Average		14.82	0.22	6.76	39.19	7.90	28.45

makespan (that is computed after solving a relaxation of Model (MILP)). We define two relative deviations: $Gap_{LB} = 100 \times \frac{C^H_{\max}(I) - LB(I)}{LB(I)}$, $Gap_{UB} = 100 \times \frac{C^H_{\max}(I) - \bar{C}_{\max}(I)}{\bar{C}_{\max}(I)}$. The mean deviations with respect to the lower bounds (obtained over 10 randomly generated instances for each problem size) as well as the CPU times are displayed in Table 2.

From Table 2, we observe that the MIP-based heuristic yields an average deviation of 6.76% while requiring an average CPU times of 39.19 s. However, the mean gap provided by the GA is 7.90% and a CPU times of 28.45 s. The results summarized above represent an improvement of 37.7% in CPU times and 16.87% in the relative deviations. The LP-based heuristic yields a larger average deviation of 14.82% but is extremely fast as it requires only 0.22 s on the average. It is worth to mention here that the average deviation is computed based on the best known lower bound and not the best upper bound or the exact solution.

In Table 3, an in-depth comparison between the heuristics is displayed. The first column from this table reports the deviations with respect to the best upper bound and the second column reports the number of instances where the heuristic yields the best solution. We see from this table that the proposed GA outperforms the other heuristics for large instances and is therefore recommended for instances with more than 40 jobs and 4 machines.

Table 3. Relative performance of the proposed heuristics.

		LP-based		*MIP-based*		*GA*	
m	n	GAP_{UB}	*Best*	GAP_{UB}	*Best*	GAP_{UB}	*Best*
3	15	9.50	0	0.00	10	3.94	0
	20	8.83	0	0.00	10	2.97	0
	30	8.64	0	0.06	9	1.40	1
	40	9.83	0	0.29	8	1.65	2
	50	9.45	0	0.36	7	1.23	3
4	10	8.01	0	0.00	10	3.29	0
	15	9.27	0	0.00	10	4.37	0
	20	8.50	0	0.19	8	2.46	2
	30	8.85	0	0.13	7	1.40	3
	40	8.14	0	2.45	1	0.21	9
	50	8.69	0	1.56	4	0.64	6
5	10	7.68	0	0.00	10	4.10	0
	15	9.20	0	0.09	7	2.68	3
	20	8.22	0	0.05	8	1.50	2
	30	8.63	0	2.42	5	1.53	5
	40	7.05	0	4.21	0	0.00	10
	50	7.70	0	3.22	0	0.00	10
Average		8.50		0.84		2.01	

6. Conclusion

In this paper, we investigated the problem of finding a (non-cyclic) schedule of a robotic cell that is composed of a number of serial machines and a material handling robot. We considered the case where the job processing times are controllable and depend on the amount of resource allocated to the processing of the operation. The problem requires finding the processing times, the job sequence, as well as the sequencing of the robot moves so that the makespan is minimized. The relevance of this objective stems from the fact that it amounts to maximizing throughput. To solve this complex scheduling problem, we proposed a valid MIP formulation that was subsequently embedded (in its original form or using some relaxations) within different heuristic approaches including a novel GA. We presented the results of a computational study that demonstrate the effectiveness of the proposed optimization-based approaches. In particular, the genetic algorithm exhibits the best performance for instances with more than 40 jobs and 5 machines.

This research addressed the case where the processing times vary linearly with the amount of allocated resource. Future research may be directed toward scheduling robotic cells with controllable processing times specified by a nonlinear convex resource consumption function. Two variants might be considered: (i) minimizing the makespan with a budget constraint or; (ii) minimizing the production cost with a restriction on the maximum makespan. These problems are both challenging and highly relevant to contemporary manufacturing systems. The development of effective solution approaches for such scheduling models is part of our ongoing research.

References

[1] M.S. Akturk and T. Ilhan, Single cnc machine scheduling with controllable processing times to minimize total weighted tardiness. *Comput. Oper. Res.* **38** (2011) 771–781.

[2] N. Al-Dhaheri and A. Diabat, The quay crane scheduling problem. *J. Manufact. Syst.* **36** (2015) 87–94.

[3] N. Al-Dhaheri, A. Jebali and A. Diabat, A simulation-based genetic algorithm approach for the quay crane scheduling under uncertainty. *Simul. Model. Pract. Theory* **66** (2016) 122–138.

[4] M. Al-Salem, M. Haouari, M. Kharbeche and W. Khallouli, A free-slack-based genetic algorithm for the robotic cell problem with controllable processing times, in: Heuristics, Metaheuristics and Approximate Methods in Planning and Scheduling. Springer (2016) 77–93.

[5] D. Biskup and T.E. Cheng, Single-machine scheduling with controllable processing times and earliness, tardiness and completion time penalties. *Engrg. Optim.* **31** (1999) 329–336.

[6] J. Carlier, M. Haouari, M. Kharbeche and A. Moukrim, An optimization-based heuristic for the robotic cell problem. *Eur. J. Oper. Res.* **202** (2010) 636–645.

[7] M.W. Dawande, H.N. Geismar, S.P. Sethi and C. Sriskandarajah. Vol. 101 of Throughput optimization in robotic cells. Springer Science & Business Media (2007).

[8] A. Diabat, Hybrid algorithm for a vendor managed inventory system in a two-echelon supply chain. *Eur. J. Oper. Res.* **238** (2014) 114–121.

[9] A. Diabat and E. Theodorou, An integrated quay crane assignment and scheduling problem. *Comput. Indus. Eng.* **73** (2014) 115–123.

[10] A. Diabat and M. Al-Salem, An integrated supply chain problem with environmental considerations. *Int. J. Prod. Econ.* **164** (2015) 330–338.

[11] A. Diabat and R. Deskoores, A hybrid genetic algorithm based heuristic for an integrated supply chain problem. *J. Manufact. Syst.* **38** (2016) 172–180.

[12] A. Diabat, O. Battaïa and D. Nazzal, An improved lagrangian relaxation-based heuristic for a joint location-inventory problem. *Comput. Oper. Res.* **61** (2015) 170–178.

[13] H. Gultekin, M.S. Akturk and O.E. Karasan, Bicriteria robotic cell scheduling. *J. Schedul.* **11** (2008) 457–473.

[14] H. Gultekin, M.S. Akturk and O.E. Karasan, Bicriteria robotic operation allocation in a flexible manufacturing cell. *Comput. Oper. Res.* **37** (2010) 779–789.

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

[16] Y.-M. Fu, A. Diabat and I.-T. Tsai, A multi-vessel quay crane assignment and scheduling problem: Formulation and heuristic solution approach. *Expert Syst. Appl.* **41** (2014) 6959–6965.

[17] A. Janiak, Single machine scheduling problem with a common deadline and resource dependent release dates. *Eur. J. Oper. Res.* **53** (1991) 317–325.

[18] S.M. Johnson, Optimal two-and three-stage production schedules with setup times included. *Nav. Res. Logist. Q.* **1** (1954) 61–68.

[19] M. Karimi-Nasab and S.F. Ghomi, Multi-objective production scheduling with controllable processing times and sequence-dependent setups for deteriorating items. *Int. J. Prod. Res.* **50** (2012) 7378–7400.

[20] V. Kayvanfar, G.M. Komaki, A. Aalaei and M. Zandieh, Minimizing total tardiness and earliness on unrelated parallel machines with controllable processing times. *Comput. Oper. Res.* **41** (2014) 31–43.

[21] M. Kharbeche, J. Carlier, M. Haouari and A. Moukrim, Exact methods for the robotic cell problem. *Flexible Ser. Manufact. J.* **23** (2011) 242–261.

[22] C. Koulamas, S. Gupta and G.J. Kyparisis, A unified analysis for the single-machine scheduling problem with controllable and non-controllable variable job processing times. *Eur. J. Oper. Res.* **205** (2010) 479–482.

[23] K. Li, Y. Shi, S.-l. Yang and B.-Y. Cheng, Parallel machine scheduling problem to minimize the makespan with resource dependent processing times. *App. Soft Comput.* **11** (2011) 5551–5557.

[24] B. Mor and G. Mosheiov, Batch scheduling of identical jobs with controllable processing times. *Comput. Oper. Res.* **41** (2014) 115–124.

[25] M. Nawaz, E.E. Enscore and I. Ham, A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega* **11** (1983) 91–95.

[26] G. Niu, S. Sun, P. Lafon, Y. Zhang and J. Wang, Two decompositions for the bicriteria job-shop scheduling problem with discretely controllable processing times. *Int. J. Prod. Res.* **50** (2012) 7415–7427.

[27] Y.-B. Park, Optimizing robot's service movement in a robot-centered fmc. *Comput. Indus. Eng.* **27** (1994) 47–50.

[28] P. Renna, Controllable processing time policies for job shop manufacturing system. *Int. J. Adv. Manufact. Technol.* **67** (2013) 2127–2136.

[29] D. Shabtay and G. Steiner, A survey of scheduling with controllable processing times. *Disc. Appl. Math.* **155** (2007) 1643–1666.

[30] D. Shabtay, M. Kaspi and G. Steiner, The no-wait two-machine flow shop scheduling problem with convex resource-dependent processing times. *IIE Trans.* **39** (2007) 539–557.

[31] H.D. Sherali and W.P. Adams, A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discrete Math.* **3** (1990) 411–430.

[32] A. Shioura, N.V. Shakhlevich and V.A. Strusevich, A submodular optimization approach to bicriteria scheduling problems with controllable processing times on parallel machines. *SIAM J. Discrete Math.* **27** (2013) 186–204.

[33] Z. Uruk, H. Gultekin and M.S. Akturk, Two-machine flowshop scheduling with flexible operations and controllable processing times. *Comput. Oper. Res.* **40** (2013) 639–653.

[34] D. Vasiljevic and M. Danilovic, Handling ties in heuristics for the permutation flow shop scheduling problem. *J. Manufact. Syst.* **35** (2015) 1–9.

[35] K. Xu, Z. Feng and K. Jun, A tabu-search algorithm for scheduling jobs with controllable processing times on a single machine to meet due-dates. *Comput. Oper. Res.* **37** (2010) 1924–1938.

[36] K. Xu, Z. Feng and L. Ke, A branch and bound algorithm for scheduling jobs with controllable processing times on a single machine to meet due dates. *Ann. Oper. Res.* **181** (2010) 303–324.

[37] K. Xu, Z. Feng and L. Ke, Single machine scheduling with total tardiness criterion and convex controllable processing times. *Ann. Oper. Res.* **186** (2011) 383–391.

[38] D.-L. Yang, T. Cheng and S.-J. Yang, Parallel-machine scheduling with controllable processing times and rate-modifying activities to minimise total cost involving total completion time and job compressions. *Int. J. Prod. Res.* **52** (2014) 1133–1141.

[39] N. Yin and X.-Y. Wang, Single-machine scheduling with controllable processing times and learning effect. *Int. J. Adv. Manufact. Technol.* **54** (2011) 743–748.

[40] Y. Yin, T. Cheng, C.-C. Wu and S.-R. Cheng, Single-machine common due-date scheduling with batch delivery costs and resource-dependent processing times. *Int. J. Prod. Res.* **51** (2013a) 5083–5099.

[41] Y. Yin, T.E. Cheng, C.-C. Wu, S.-R. Cheng, Single-machine due window assignment and scheduling with a common flow allowance and controllable job processing time. *J. Oper. Res. Soc.* **65** (2013b) 1–13.

[42] S. Yildiz, M.S. Akturk and O.E. Karasan, Bicriteria robotic cell scheduling with controllable processing times. *Int. J. Prod. Res.* **49** (2011) 569–583.

[43] S. Zdrzałka, Scheduling jobs on a single machine with release dates, delivery times and controllable processing times: worst-case analysis. *Oper. Res. Lett.* **10** (1991) 519–523.

[44] Q. Zeng, A. Diabat and Q. Zhang, A simulation optimization approach for solving the dual-cycling problem in container terminals. *Maritime Policy Manage.* **42** (2015) 806–826.