Exact Method for Robotic Cell Problem

Mohamed Kharbeche\textsuperscript{1}, Jacques Carlier\textsuperscript{2}

\textit{UMR CNRS 6599 Heudiasyc, Centre de Recherches de Royallieu}
\textit{Université de Technologie de Compiègne}
\textit{France}

Mohamed Haouari\textsuperscript{3}

\textit{Department of Industrial and Systems Engineering}
\textit{Faculty of Engineering Ozyegin University}
\textit{Istanbul, Turkey}

ROI - Combinatorial Optimization Research Group
\textit{Ecole Polytechnique de Tunisie}
\textit{La Marsa, Tunisie}

Aziz Moukrim\textsuperscript{4}

\textit{UMR CNRS 6599 Heudiasyc, Centre de Recherches de Royallieu}
\textit{Université de Technologie de Compiègne}
\textit{France}

Abstract

This study investigates an exact method for the \textit{Robotic Cell Problem}. We present an exact branch and bound algorithm which is the first exact procedure specifically designed for this strongly NP-hard problem. In this paper, we propose a new mathematical formulation and we describe a new lower bound for the RCP. In addition, we propose a genetic algorithm. We report that the branch and bound algorithm is more effective than the proposed mathematical formulation which can
solve small sized problem. Also, computational study provides evidence that the genetic algorithm delivers reasonably good solutions while requiring significantly shorter CPU times to solve this problem.

Keywords: Robotic Cell, Mathematical formulation, GA, Branch and Bound.

1 Introduction

The Robotic Cell Problem (RCP) is 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 $M_{m+1}$. 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 $\tau_{ih}$ 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. 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, such that the time $C_{\text{max}}$ (makespan) at which all the jobs are completed 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[4]). Hall and Sriskandarajah[3] 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.

This problem has been lately studied by Carlier et al.[2]. We refer to this paper for the literature review. The main contribution of this paper, is to propose exact and approximate methods for the RCP. The remainder of this paper is organized as follows. In Section 2, we present a new mathematical
formulation and a new lower bound for the RCP. The genetic algorithm and the branch and bound algorithm are presented in section 3. The empirical performance of the proposed approach is assessed in Section 4. Finally, we provide in Section 5 some concluding remarks.

2 Solution of the RCP

In this section, we describe a mixed-integer programming model for RCP which can solve to optimality small sized problems. Also, we describe an improved lower bound that is derived from Carlier et al.[2].

2.1 Mathematical formulation for the RCP

To the best of our knowledge, there is no mathematical model dealing with the RCP in the literature. The aim of this mixed-integer linear program is to find the schedule as well as the sequence of robot moves that minimizes the completion time. This new model is described as follows:

Let: \( x_{kj} = 1 \) if job \( j \) is assigned at position \( k \), and 0 otherwise, \( \forall \ j, k = 1, ..., n \),
\( t_{ik} \) : the starting time of the transfer from machine \( M_i \) to \( M_{i+1} \) of job assigned at position \( k \), \( \forall \ k = 1, ..., n \) and \( i = 0, ..., m \)
\( y_{ih} = 1 \) if operation \( h \in H \) is performed by the robot after operation \( l \) and 0 otherwise, \( \forall \ l < h \) (where the notation “\( l < h \)” means that operation \( l \) precedes operation \( h \)) and \( H \) is the set of robot operations \( \{O_{ik} : i = 0, ..., m, \ k = 1, ..., n\} \).

The mixed-integer linear program can be written as follows:

\[
\text{Minimize } t_{mn}
\]  

Subject to:

\[
\sum_{k=1}^{n} x_{kj} = 1, \ \forall \ j = 1, ..., n,
\]  

\[
\sum_{j=1}^{n} x_{kj} = 1, \ \forall \ k = 1, ..., n,
\]

\[
t_{i+1,k} \geq t_{ik} + \sum_{j=1}^{n} p_{i+1,j} x_{kj} + \tau_{i,i+1} \ \forall k = 1, ..., n \ \forall i = 0, ..., m - 1,
\]

\[
t_{i-1,k} \geq t_{i,k-1} + \tau_{i,i+1} + \tau_{i+1,i-1} \ \forall k = 2, ..., n, \ \forall i = 1, ..., m
\]
\[ \sum_{l: l \prec h} y_{lh} = 1, \ \forall \ h \in H \quad (6) \]

\[ \sum_{i: h \prec i} y_{hl} = 1, \ \forall \ h \in H \quad (7) \]

\[ t_{i-1,k} \geq t_{i+1,k-1} + \tau_{i+1,i+2} + \tau_{i+2,i-1} + M(yhl - 1), \quad \forall k = 2, \ldots, n, \ \forall i = 1, \ldots, m - 1, \ l \equiv (i - 1, k), \ h \equiv (i + 1, k - 1) \quad (8) \]

\[ t_{i+1,k-1} \geq t_{i-1,k-1} + \tau_{i-1,i} + \tau_{i,i+1} + M(yhl - 1), \quad \forall k = 2, \ldots, n, \ \forall i = 1, \ldots, m - 1, \ l \equiv (i + 1, k - 1), \ h \equiv (i - 1, k) \quad (9) \]

\[ t_{0k} \geq t_{m,k-p} + \tau_{m,m+1} + \tau_{m+1,0} + M(yhl - 1), \quad \forall k = 2, \ldots, n, \ \forall p = 1, \ldots, m - 1, \ l \equiv (0, k), \ h \equiv (m, k - p) \quad (10) \]

\[ t_{m,k-1} \geq t_{0,k+p} + \tau_{01} + \tau_{1m} + M(yhl - 1), \quad \forall k = 2, \ldots, n, \ \forall p = 1, \ldots, m - 1, \ l \equiv (0, k + p), \ h \equiv (m, k - 1) \quad (11) \]

\[ t_{ik} \geq 0, \ \forall \ k = 1, \ldots, n, \ \forall i = 0, \ldots, m \quad (12) \]

\[ x_{kj} \in \{0, 1\}, \ \forall \ j,k = 1, \ldots, n, \ \text{and} \ y_{lh} \in \{0, 1\}, \ \forall \ l < h. \quad (13) \]

The objective function (1) minimizes the beginning of the last operation which is the robot transfer of job scheduled at the last position from the last machine \( M_m \) to the Output \( M_{m+1} \). Constraints (2) and (3) require that job must be scheduled at one position and at each position there is only one job scheduled, respectively. Constraints (4) are the precedence constraints of operations that concern the same job. Constraints (5), means that after unloading the job at position \( k - 1 \) on machine \( M_{i+1} \), the robot has to move to machine \( M_{i-1} \) to upload the job at position \( k \) and transfer it to machine \( M_i \). Constraints (6) and (7) require that each operation has exactly one successor and one predecessor, respectively. Constraints (8)-(11) ensure that the precedence constraints between particular operations must be satisfied. Finally, Constraints (12)-(13) represent binary restrictions on the \( x \)-variables and the \( y \)-variables, respectively.

### 2.2 A new lower bound for the RCP

The point behind this improved lower bound is to schedule a job at the first position and another job at the last position. The case of two machines is solved by Aneja and Kamoun [1]. 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 subject to heads and tails, where we define

- a head of job \(j\) scheduled at the first position \(\tilde{r}_j = \sum_{k=1}^{i} p_{kj} + \sum_{k=0}^{i-1} \tau_{k,k+1}, \forall j = 1, ..., n\)

- a tail of job \(l\) scheduled at the last position \(\tilde{q}_l = \sum_{k=i+1}^{m} p_{kl} + \sum_{k=i}^{m} \tau_{k,k+1}, \forall l = 1, ..., n\)

We must evaluate \(n(n-1)\) combinations where job \(j\) is scheduled at the first position and job \(l \neq j\) at the last position. Then a valid lower bound is:

\[
LB = \max_{1 \leq i \leq m-1} \{ \min_{1 \leq j, 1 \leq l \leq n} \{ \tilde{r}_j + LB^i + \tilde{q}_l \} \} \tag{14}
\]

where \(LB^i\) is the exact solution of the RCP with two machines using the transformation in Carlier et al. [2]. \(LB\) can be computed in \(O(mn^3)\)-time.

3 A branch and bound algorithm for the RCP

In this section, we describe the main feature of the genetic algorithm as well as the exact branch and bound for the RCP.

3.1 Computing an upper bound using genetic algorithms

We have developed three genetic algorithms. The main feature are of the GA:

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

- **The crossover operator**: The SJOX Similar Job Order Crossover of Ruiz et al.[5].

- **The mutation operator**: The pairwise interchange.

- **Fitness computation**: In \(GA1\) the sequence of the robot moves is obtained using the a list scheduling algorithm: In \(GA2\), we suppose that the robot repeats the same moves for all jobs. Finally in \(GA3\), we determined a chromosome for the \(F|\text{Block}, t_k|C_{\max}\) then we computed an upper bound for the RCP using a mixed-integer linear program that gives optimal robot moves. Next, the performance of \(GA\) will be compared against the performance of the proposed exact methods.
3.2 Synthesis of the branch and bound algorithm

In order to start with a good upper bound we used a starting solution obtained from the genetic algorithms. We sort the jobs according to the GA solution. The main features of the branch and bound algorithm that we have implemented for finding an optimal sequence of the RCP are the following:

- **Root node**: The root node corresponds to a partial schedule obtained from the best solution of genetic algorithm.

- **Branching scheme**: We sequenced the job at the first available position.

- **Branching strategy**: We employed a depth-first search strategy. The advantages of a depth-first strategy are: the number of active nodes is less than the number of jobs, and the bottom of the tree is reached faster so a feasible solution can be found earlier.

- **Lower bound**: For each newly created node, we compute the lower bound described in section 2.2

- **Upper bound**: An upper bound is computed if a feasible schedule is obtained using a modified linear formulation that gives the optimal moves of robot. This formulation is the same as the formulation in section 2.1 but does not contain the $x_{kj}$ variables because the sequence of jobs is known in advance.

- **Search strategy**: We select the node that has as an unscheduled job the first one according to the root sequence.

4 Computational results

The test-bed was generated in the following way. The processing times are drawn from the discrete uniform distribution on $[1, 50]$ and the transportation time between a pair of machines $M_i$ and $M_k$ is $\tau_{i,k} = 2 \times |i - k|$. The number of jobs $n$ is taken equal to $n = 10, 12, 14, 16, 18, \text{ and } 20$. The number of machines $m$ is taken equal to 3, 4, and 5. For each $(m,n)$ combination, 10 instances were randomly generated. The proposed algorithms were coded and compiled with Microsoft Visual Studio C++ (2005). The mathematical models have been solved using CPLEX 11.1. All the computational experiments were carried out on a Pentium IV 3.3 GHz, 3 Gbytes RAM PC. In Table below, 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

- **UNS1, UNS2**: number of unsolved instances after reaching a 1-hour time limit of the MILP respectively of the branch and bound.
<table>
<thead>
<tr>
<th>(m,n)</th>
<th>TNO</th>
<th>UNS1</th>
<th>UNS2</th>
<th>Time 1</th>
<th>Time 2</th>
<th>Time GA</th>
<th>Gap LB</th>
<th>Gap GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3, 10)</td>
<td>70</td>
<td>0</td>
<td>0</td>
<td>1.66</td>
<td>0.12</td>
<td>11.44</td>
<td>3.84</td>
<td>0.02</td>
</tr>
<tr>
<td>(3, 12)</td>
<td>84</td>
<td>0</td>
<td>0</td>
<td>276.38</td>
<td>3.83</td>
<td>14.90</td>
<td>3.97</td>
<td>0.22</td>
</tr>
<tr>
<td>(3, 14)</td>
<td>98</td>
<td>1</td>
<td>0</td>
<td>362.56</td>
<td>31.24</td>
<td>18.24</td>
<td>4.42</td>
<td>0.16</td>
</tr>
<tr>
<td>(3, 16)</td>
<td>112</td>
<td>3</td>
<td>0</td>
<td>896.80</td>
<td>427.55</td>
<td>22.74</td>
<td>3.07</td>
<td>0.18</td>
</tr>
<tr>
<td>(3, 18)</td>
<td>126</td>
<td>9</td>
<td>4</td>
<td>0.14</td>
<td>882.09</td>
<td>29.34</td>
<td>3.36</td>
<td>0.17</td>
</tr>
<tr>
<td>(3, 20)</td>
<td>140</td>
<td>10</td>
<td>9</td>
<td>*</td>
<td>318.31</td>
<td>32.56</td>
<td>3.29</td>
<td>0.00</td>
</tr>
<tr>
<td>(4, 10)</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>6.87</td>
<td>0.23</td>
<td>14.70</td>
<td>8.58</td>
<td>0.25</td>
</tr>
<tr>
<td>(4, 12)</td>
<td>108</td>
<td>0</td>
<td>0</td>
<td>225.90</td>
<td>3.60</td>
<td>19.22</td>
<td>7.64</td>
<td>0.55</td>
</tr>
<tr>
<td>(4, 14)</td>
<td>144</td>
<td>4</td>
<td>0</td>
<td>1115.72</td>
<td>110.88</td>
<td>24.36</td>
<td>7.73</td>
<td>0.87</td>
</tr>
<tr>
<td>(4, 16)</td>
<td>153</td>
<td>10</td>
<td>3</td>
<td>*</td>
<td>1262.42</td>
<td>31.50</td>
<td>7.61</td>
<td>0.67</td>
</tr>
<tr>
<td>(4, 18)</td>
<td>162</td>
<td>10</td>
<td>10</td>
<td>*</td>
<td>*</td>
<td>46.44</td>
<td>8.52</td>
<td>*</td>
</tr>
<tr>
<td>(4, 20)</td>
<td>180</td>
<td>10</td>
<td>10</td>
<td>*</td>
<td>*</td>
<td>58.15</td>
<td>9.27</td>
<td>*</td>
</tr>
<tr>
<td>(5, 10)</td>
<td>110</td>
<td>0</td>
<td>0</td>
<td>343.38</td>
<td>10.53</td>
<td>23.44</td>
<td>10.80</td>
<td>0.97</td>
</tr>
<tr>
<td>(5, 12)</td>
<td>132</td>
<td>4</td>
<td>0</td>
<td>884.66</td>
<td>132.91</td>
<td>32.63</td>
<td>9.60</td>
<td>1.03</td>
</tr>
<tr>
<td>(5, 14)</td>
<td>176</td>
<td>9</td>
<td>4</td>
<td>954.97</td>
<td>918.00</td>
<td>53.35</td>
<td>11.90</td>
<td>1.04</td>
</tr>
<tr>
<td>(5, 16)</td>
<td>176</td>
<td>10</td>
<td>9</td>
<td>*</td>
<td>2669.97</td>
<td>92.10</td>
<td>12.32</td>
<td>0.40</td>
</tr>
<tr>
<td>(5, 18)</td>
<td>198</td>
<td>10</td>
<td>10</td>
<td>*</td>
<td>*</td>
<td>65.76</td>
<td>12.05</td>
<td>*</td>
</tr>
<tr>
<td>(5, 20)</td>
<td>220</td>
<td>10</td>
<td>10</td>
<td>*</td>
<td>*</td>
<td>79.01</td>
<td>13.10</td>
<td>*</td>
</tr>
</tbody>
</table>

Table 1
Performance of the different approaches

- **Time 1**, **Time 2**: mean CPU time of the LP respectively of the branch and bound
- **Time GA**: mean CPU time of the genetic algorithms
- **GAP LB**: The average percentage deviation of the lower bound from the value $UB_{GA}$ of the best solution of $GA$ (i.e. $100 \times (UB_{GA} - LB)/UB_{GA}$)
- **GAP GA**: The average percentage deviation of the best GA from the exact value (i.e. $100 \times (UB_{GA} - Opt)/Opt$) where $Opt$ is the optimal value.
We have noticed from this table that the mathematical formulation can solve small sized instances with 16 jobs and 3 machines, 14 jobs and 4 machines and 12 jobs and 5 machines. Nevertheless, this is a remarkable result since the flow shop problems with blocking are notoriously hard to solve to optimality. We have found that the branch and bound algorithm can solve larger problem instances with 18 jobs and 3 machines, 16 jobs and 4 machines and 14 jobs and 5 machines. Also, the mean CPU time of the branch and bound algorithm is less than the one of the mixed-integer programming formulation. Interestingly, we have observed that the average percentage deviation of the GA from the optimal solution is less than 1.05%.

5 Conclusion

In this paper, we have investigated the robotic cell problem. We have proposed exact and approximate methods for this problem. To the best of our knowledge, this is the first attempt to solve to optimality this complex scheduling problem. Also, we have described a new linear model and a new lower bound for the RCP. We reported the results of a computational study that provide that the proposed mathematical model can solve small problems. However, the branch and bound algorithm can solve larger problem instances. In addition, we have found that the genetic algorithm requires significantly shorter CPU times and is proven useful for efficiently solving the RCP problem.

References


