Integrating Scheduling and Distribution: Algorithms and Insights

Roberto Fernandes Tavares Neto
Universidade Federal de São Carlos
Rod. Washington Luís - Km 235 - São Carlos - SP - Brazil
tavares@dep.ufscar.br

ABSTRACT

The integration between production and distribution decisions has been proven as an important topic for the industry and a significant research challenge. Although there are some noteworthy results on this field, there is a need for simple algorithms that can be adapted in different problem structures. This paper approaches an integrated scheduling-distribution problem (ISDP), where the production of goals is performed by a set of parallel machines and the distribution is a variation of the well-known single vehicle routing problem with multiple routes. The constructive algorithm is expanded into two variations of the Iterated Greedy procedure, where two destruction policies are compared. Those three approaches were validated using a mathematical model and then their behavior is analyzed on the solution of larger instances. In our results, we show that: (i) the IG can be used to improve the solution given by the constructive procedure; (ii) the destruction procedure based on random job removal generated superior results than the one based on route removal; (iii) although different generation schemes related to the setup or distances influences the behavior of both IG algorithms, changing the range of the order’s size does not generate significant impact on the final relative result.

KEYWORDS. Integrated Scheduling Distribution Problems, Scheduling, VRP

AD&GP - PO na Administração e Gestão da Produção
L&T - Logística e Transportes
MH - Metaheuristicas
1. Introduction

According to [Wang et al., 2015], integrated scheduling-distribution models can be classified into two groups: (i) problems that considers the integration between production and transportation decisions and (ii) problems that considers the integration between production, inventory and distribution decisions. On the first group, named by the authors as Integrated Scheduling of Production and Distribution Problems (ISPDP), the finished product inventory is negligible, and occurs mainly on time-sensitive production (such as newspapers ans some chemical products). Some authors (e.g. Gao et al. [2015]) refers to the non-existence of this inventory as a no-wait constrain between production and distribution.

Still adopting the notation given by [Wang et al., 2015], the second group considers a finished product inventory linking the production and the distribution of the system. This inventory is mainly used to soften the impacts of the different production/consumption rates found in the system. The problems of this second group are labeled as Integrated Scheduling od Production, Inventory and Distribution Problems (ISPIDP).

The production stage of an integrated problem is modeled in the literature as different structures. Some authors, such as [Gao et al., 2015], assumes that the orders are processed (and delivered) in batches. The production section is also assumed as a single machine environment in works such as [Zhong et al., 2010] and [Ng e Lu, 2012].

An examination of the current literature, brings us some different approaches to model the distribution part of the ISDP: some authors, such as [Cheng et al., 2015], assumes that the route costs are the same, regardless the number of clients to be attend. Although dedicated routes are also common (e.g. [Ng e Lu, 2012]), [Chen, 2010] states that the most challenging distribution structure is when there is a need to establish the delivery sequence of route, since the problem became an extension of the well-known NP-Hard Vehicle Routing Problem. A significant number of the variations of the routing problem of the ISDP can be found in the literature, ranging from a single vehicle (e.g. [Gao et al., 2015]) to multiple vehicles (e.g. [Cheng et al., 2015]).

This paper approaches the ISD problem where the manufacturing environment is represented by a parallel machine environment, and the distribution is performed by a single capacitated vehicle with multiple routes. The goal is to minimize the system makespan. A mathematical model, a constructive heuristic and two iterated greedy-based heuristics are proposed, implemented and analyzed.

The remaining of this paper is organized as follows: section 2 presents the mathematical model; section 3 presents the constructive heuristic used as a basis for the ig-based algorithms proposed in section 4. The results are presented in section 5 and the final remarks are discussed in section 6.

2. Problem Definition

This paper approaches an ISPIDP described as follows: the manufacturing is either a single machine or a set of 2 or 3 identical parallel machines. There is no constrain regarding the finished product inventory. The distribution is performed by a single vehicle with limited capacity whose routes must be determined assuming a VRP-based structure. The symbols used to define a problem instance are given in Table 1. To mathematically define this problem, section 2.1 presents a Mixed-Integer formulation of this problem.

2.1. A MIP approach

To represent this problem using a MIP formulation, this research uses the variables presented in table 2. To present the model, section 2.1.1 presents the equations used to assure the production schedule structure represented by the $x_{ij}$ variables; the constrains regarding to the completion time of each order at manufacturing are presented in section 2.1.2; the viability of the routes represented by $w_{ijk}$ is assured by the constrains presented in section 2.1.3; section 2.1.4 presents...
Table 1: Symbols Used to Define a Problem Instance

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i, j$</td>
<td>Indexes used to identify processing order</td>
</tr>
<tr>
<td>$f$</td>
<td>Index used to identify machines</td>
</tr>
<tr>
<td>$k$</td>
<td>Index used to identify a route</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Number of machines</td>
</tr>
<tr>
<td>$\rho_i$</td>
<td>The processing time of order $i$</td>
</tr>
<tr>
<td>$\kappa_{ij}$</td>
<td>The setup time required to switch production from order $i$ to order $j$</td>
</tr>
<tr>
<td>$\sigma_i$</td>
<td>The size of $i$</td>
</tr>
<tr>
<td>$\psi$</td>
<td>The total capacity of the vehicle</td>
</tr>
<tr>
<td>$\delta_{ij}$</td>
<td>The distance between $i$ and $j$</td>
</tr>
</tbody>
</table>

Table 2: Symbols Used to Define a Problem Instance

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{ipf}$</td>
<td>Binary</td>
<td>1 if job $i$ is produced at position $p$ of machine $f$; 0 otherwise</td>
</tr>
<tr>
<td>$w_{ijk}$</td>
<td>Binary</td>
<td>1 if job $j$ is delivery by route $k$ immediately after job $j$; 0 otherwise</td>
</tr>
</tbody>
</table>

2.1.1. Constrains related to $x$ variables

\[
x_{0pf} = 0 \quad \forall \left\{ \begin{array}{l} p \\ f \end{array} \right. \tag{1}
\]
\[
\sum_{p,f} x_{ipf} = 1 \quad \forall i > 0 \tag{2}
\]
\[
\sum_i x_{ipf} \leq 1 \quad \forall \left\{ \begin{array}{l} p \\ f \end{array} \right. \tag{3}
\]
\[
\sum_j x_{j(p+1)f} \leq \sum_i x_{ipf} \quad \forall \left\{ \begin{array}{l} p < |P| \\ f \end{array} \right. \tag{4}
\]

Where equation 1 assures that dummy order 0 will not be scheduled on any position other than position 0; Equation 2 forces an order to be produced only once in all machines/position; Equation 3 assures that a position of a machine holds at most one order; Equation 4 assures that a sequence of any machine will have unused positions only after the used positions.
2.1.2. Constrains related to C variables

\[ C_0 = 0 \]  \hspace{1cm} (5)  
\[ GP = \sum_i \rho_i + \sum_{i,j} \kappa_{ij} \]  \hspace{1cm} (6)  
\[ C_i \geq \rho_i + \kappa_{0i} - GP \cdot (1 - x_{i0}) \]  \hspace{1cm} \forall \{ i \} \]  \hspace{1cm} (7)  
\[ C_j \geq C_i + \rho_j + \kappa_{ij} - GP \cdot (2 - x_{ipf} - x_{j(p+1)f}) \]  \hspace{1cm} \forall \{ j > 0 \} \]  \hspace{1cm} \forall \{ i > 0 \} \]  \hspace{1cm} \forall \{ i \neq j \} \]  \hspace{1cm} \forall \{ p < |P| \} \]  \hspace{1cm} (8)  

Where Equation 5 sets the completion time of the dummy job; Equations 6 defines the value of the "Big-M" constant used in the next constrains; Equations 7 set the completion time of the order allocated on the first sequence of each machine; Equations 8 set the completion times of the remaining positions.

2.1.3. Constrains related to w variables

\[ w_{00(k-1)} \leq w_{00k} \]  \hspace{1cm} \forall \{ k \} \]  \hspace{1cm} (10)  
\[ \sum_k w_{ipk} \leq 1 \]  \hspace{1cm} \forall \{ i > 0 \} \]  \hspace{1cm} \forall \{ j > 0 \} \]  \hspace{1cm} (11)  
\[ \sum_{i,k} w_{ijk} \leq 1 \]  \hspace{1cm} \forall \{ p \} \]  \hspace{1cm} \forall \{ f \} \]  \hspace{1cm} (12)  
\[ \sum_j w_{0jk} = 1 \]  \hspace{1cm} \forall \{ k \} \]  \hspace{1cm} (13)  
\[ \sum_j w_{j0k} = 1 \]  \hspace{1cm} \forall \{ k \} \]  \hspace{1cm} (14)  
\[ w_{jyk} = 0 \]  \hspace{1cm} \forall \{ j > 0 \} \]  \hspace{1cm} \forall \{ k \} \]  \hspace{1cm} (15)  
\[ w_{00k} + \sum_{j>0} w_{ijk} \leq 1 \]  \hspace{1cm} \forall \{ i > 0 \} \]  \hspace{1cm} \forall \{ k \} \]  \hspace{1cm} (16)  
\[ \sum_{h \neq i} w_{ihk} = \sum_{h \neq i} w_{hik} \]  \hspace{1cm} \forall \{ i \} \]  \hspace{1cm} \forall \{ k \} \]  \hspace{1cm} (17)  

Where Equations 10 assures that there’s no empty route between two populated routes; Equations 11 assures that a link between two cities is used at most once by any vehicle; Equations 12 assures that only one route visits a city; Equations 13 and 14 allows a route to leave and arrive just once from/to the depot; Equations 15 forbids a vehicle to leave and arrive to the same city; Equations 16 forces \( w_{00k} = 0 \) when route \( k \) is used; Equations 17 maintain the network flow.
2.1.4. Constrains related to vehicle capacity

\[ A_i = 0 \] (19)
\[ G\sigma = \sum \sigma_i \] (20)
\[ A_i \leq \sigma_i \quad \forall \{i > 0 \} \] (21)
\[ A_i \geq A_j + \sigma_i - G\sigma \cdot (1 - w_{ijk}) \quad \forall \{ j \neq i \} \] (22)
\[ A_i \leq \sum_{j>0} \sum_h \sigma_j \cdot w_{jhh} + G\sigma \cdot (1 - w_{ihh}) \quad \forall \{ i > 0 \} \] (23)

Where Equation 19 sets the value of the cargo at dummy node; Equation 20 calculates the value of the "Big-M" constant; Equations 21 establish the upper bound of the vehicle usage; Equations 22 and 23 establishes the lower and upper bounds for the cargo of the vehicle. Note that Equations 19, 22 and 23 are very similar to the MTZ subtour elimination constrains.

2.1.5. Integration-related constrains

\[ R_k \geq C_i - GP \cdot (1 - w_{ijk}) \quad \forall \{ i > 0 \} \] (24)
\[ D_i \geq R_k + \delta_{i0} - GP \cdot (1 - w_{i0k}) \quad \forall \{ i > 0 \} \] (25)
\[ D_i \geq D_i + \delta_{ij} - GP \cdot (1 - \sum_k w_{ijk}) \quad \forall \{ j \neq i \} \] (26)
\[ R_k \geq D_i + \delta_{i0} - GP \cdot (1 - w_{i0(k-1)}) \quad \forall \{ k > 0 \} \] (27)

Where equations 24 assures that a route \( k \) only depart after all the orders delivered by this route are produced; Equations 25 set the delivery time of the orders located in the first position of the routes; Equations 26 set the delivery time of the remaining orders of the route; Equations 27 establishes that a route \( k \) can only depart after route \( k - 1 \) arrives.

2.1.6. Fitness function

\[ z \geq D_i + \delta_{i0} - M \cdot (1 - x_{i0k}) \quad \forall \{ i > 0 \} \] (28)

As mentioned before, the objective is to minimize the makespan. In our case, the system makespan is when the last resource became available - and that happens when the last vehicle returns from the last delivery. By minimizing \( z \) given by equations 28, one can minimize the system makespan.

3. A constructive heuristic

Unfortunately, our literature review could not find any suitable constructive algorithm to solve this problem. Since a constructive algorithm is usually a basis for any further development of more complex solution procedures, the first step of our research was to derive a heuristic based on the well-known NEH algorithm ([Nawaz et al., 1983]). This algorithm is an extension of the algorithms for single machine manufacturing environments presented by [Tavares Neto e Asano, 2015], [Tavares Neto e Ogawa, 2015] and [Tavares Neto e Oliveira, 2015]. As presented in algorithm 1, the
Algorithm 1: The pseudo-code for the constructive heuristic

\[
J \leftarrow \text{set of jobs ordered by some criteria;}
S_\pi, S_\pi^* \leftarrow \text{manufacturing sequences;}
V_\pi, V_\pi^* \leftarrow \text{delivery sequences;}
S_\pi = S_\pi^* = V_\pi = V_\pi^* = \{0\};
nRoutes = 0
\]

\begin{algorithm}
\begin{algorithmic}
\For{$ps = 0 \ldots |S_\pi|}$
    \For{existing machine $m$}
        \For{position $p$ of the sequence of machine $m$}
            Insert $j$ in position $p$ of machine $m$ at $S_\pi$;
        \EndFor
    \EndFor
    \For{existing route $r$}
        \For{$pv \leftarrow \text{all positions of } r$}
            Insert $j$ at position $pv$ of route $r$, shifting the jobs on the set if necessary;
            Update the values of the best fitness $Fitness^*$ and the best sequences $S_\pi^*, V_\pi^*$.
        \EndFor
        Create a new route with only the job $j$ and insert it as last route, between each existing routes and as first route. If required, update the best values found so far. $S_\pi = S_\pi^* V_\pi = V_\pi^*$
    \EndFor
\EndFor
\end{algorithmic}
\end{algorithm}

return $S_\pi^*, V_\pi^*$

Algorithm proposed on this section inserts each job on each available position on both manufacturing and delivery sequences. The best partial solution is stored and the next order is processed using the same insertion algorithm. This cycle is executed until there is no further unprocessed orders.

A natural evolution of this constructive heuristic is to allow it to improve the solution found by some local search method. This research follows the findings of [Ruiz e Stützle, 2007] and [Ruiz e Stützle, 2008], and implements an Iterated Greedy algorithm. This algorithm is presented in section 4.

4. Improving the solution by the Iterated Greedy method

To improve the solution given by the constructive heuristic described in section 3, two Iterated Greedy algorithm were developed based on the works of Ruiz e Stützle [2008]. In this research, the authors uses the destruction-reconstruction cycle to improve the results of the NEH algorithm when this is applied to a flowshop scheduling problem. In our approach, we derive an improvement method based on those findings.

As presented in the literature, an Iterated Greedy procedure can be described based on three main routines:

Generation of an Initial Solution: Based on a standard constructive heuristic, this routine creates a first solution to be improved afterwards. Ruiz e Stützle [2008] uses the NEH algorithm. The present research uses the algorithm presented in section 3.

Destruction phase: This solution removes some jobs from the current solution. Usually, this removal is performed based on a random operator (e.g., Ruiz e Stützle [2008]). Similarly, one of the algorithms presented in this paper removes random orders from both the production and distribution sections.

Reconstruction phase: Given the partial solution remained and the removed jobs obtained from the destruction phase, the reconstruction phase applies a constructive procedure to generate a
new solution. In the case of the algorithm proposed in this section, the construction procedure is the same insertion procedure used on section 3.

This algorithm requires the specification of three parameters:

- The **ordering procedure** used by the initialization procedure: the possibilities considered were the rules mentioned in section 3 - FIFO, SPT, LPT, NNSETUP, NNDIST, SIZE, SIZEDEC.

- The **number of jobs** to be destroyed on the destruction phase. The possibilities considered were 20%, 40%, 60%, 80% and 100% of the total number of orders.

- The **number of cycles** of destruction-reconstruction performed by the algorithm. In our case, we consider \( \{1, 5, 10, 20\} \times \) number of orders.

Following those directives, two different algorithms were implemented:

**Algorithm IG** Is a standard IG algorithm, where the destruction phase removes a fixed number of random jobs of the current solution;

**Algorithm IGR** This algorithm selects a random route, and removes all jobs of that route. This step continues until the target number of jobs to be removed is achieved.

To find a suitable combination of tuning choices, a set of randomly selected instances were selected \(^1\). Then, the IRACE package (López-Ibáñez et al. [2011]) were used to find the best parameter combination. This result on the following parametrization settings: initialization by NNSETUP; removing 20% of the orders and performing a number of cycles equal to the number of orders of the instance. This setting were found to be suitable to both algorithms.

### 5. Results and Discussion

#### 5.1. Instance Generation

To analyze the performance of the algorithms, we adopt a set of 9,000 instances generated by [Tavares Neto e Oliveira, 2015]. Those instances were generated using the following procedure: the number of jobs at each instance is \( n = \{5, 10, 20, 40, 80\} \). A uniform random function was used to determine the values of the following parameters: \( \rho_i = \{1, 100\} \), \( \sigma_i = \{1, 10\} \) and \( \psi = \{\text{max}(\sigma_i); 5 \cdot \text{max}(\sigma_i)\} \). \( \kappa_{ij} \) and \( \delta_{ij} \) were determined by random positioning of points at spaces of size \( \theta_s \times \theta_s \) and \( \theta_d = \{10, 20, 30\} \), respectively. The distance between the delivery points and the production site is multiplied by \( \theta_g = \{1, 10, 20, 30\} \). With no loss of generality, we limit those values to be integers.

#### 5.2. Analysis overview

To analyze the algorithms presented in this paper, the following steps were taken:

1. To analyze the distance between solutions found by the heuristics and the optimal solution, the MIP model were implemented using Python and the CPLEX libraries provided by IBM. This analysis is presented in section 5.3

2. Unfortunately, the MIP approach could not solve problems with mode than 5 orders. As a consequence, problems of higher order were analyzed only applying the proposed heuristics. The constructive heuristic presented in section 3 is initially compared with both IG-based methods, resulting in the analysis found in section 5.4.

3. Finally, section 5.5 focuses on the comparison between the proposed IG algorithms.

\(^1\) More information about the instances can be found in section 5
5.3. Analysis of small-sized instances

To validate and analyze the difference between the optimum values found by the MIP approach and the heuristic approaches, all the problem instances of size $n = 5$ were solved by all methods. Then, for every value, the gap $= (\text{Sol FOUND} - \text{Sol BEST}) / \text{Sol BEST}$ value was calculated. Figures 1, 2 and 3 present the boxplots of the results, considering one, two and three machines respectively. In those figures, one can note that both the IG algorithms could solve the problem better than the constructive heuristic alone. Moreover, there is a clear evidence that the algorithm have found more difficulty in solve problems with more parallel machines.

5.4. Analysis of all heuristics

The second stage of our analysis seeks to analyze if the IG-based algorithms could improve significantly the base (constructive) heuristic when used to solve larger problem instances. To perform this analysis, all problems were solved by the three methods and the gap were calculated. Table 3 presents the average values found. According those values, it was clear that both the IG approaches overcome the constructive heuristic in all groups of problem instances.

5.5. Analysis of the IG heuristics

Although the difference between the constructive heuristic and the IG algorithms was clear, the same does not occur when one focuses only on the results bring by the IG-based algorithms. To analyze if the destruction strategy given by IG could provide better results than IGR, it was necessary to adopt a Wilcoxon test on the gaps found by each one of those two algorithms. In this section, we present those analysis.

In the first analysis (presented in table 4) the problem instances were grouped according the size and number of machines. The data presented shows that the destruction rule adopted by IG is clearly more suitable on problems with few parallel machines.

The second analysis performed regards the structure of the physical location of the clients, represented by parameter $\delta_{ij}$. Two analysis were performed: Table 5, shows that the IG algorithm

---

2Each group contains 81 subgroups of instances.
Table 3: Average values of gap found by each algorithm

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>NNSETUP</th>
<th>IG</th>
<th>IGR</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>0.06401</td>
<td>0.01243</td>
<td>0.02178</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0.06123</td>
<td>0.02006</td>
<td>0.01641</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>0.08299</td>
<td><strong>0.03245</strong></td>
<td>0.03881</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.04775</td>
<td>0.03818</td>
<td><strong>0.00419</strong></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0.04763</td>
<td>0.03845</td>
<td><strong>0.00382</strong></td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>0.34837</td>
<td><strong>0.00737</strong></td>
<td>0.29045</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0.06714</td>
<td>0.06385</td>
<td><strong>0.00128</strong></td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>0.06131</td>
<td>0.05808</td>
<td><strong>0.00132</strong></td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>1.29272</td>
<td><strong>0.00096</strong></td>
<td>1.16806</td>
</tr>
<tr>
<td>40</td>
<td>1</td>
<td>0.09751</td>
<td>0.09649</td>
<td><strong>0.00035</strong></td>
</tr>
<tr>
<td>40</td>
<td>2</td>
<td>0.09979</td>
<td>0.09862</td>
<td><strong>0.00030</strong></td>
</tr>
<tr>
<td>40</td>
<td>3</td>
<td>3.13046</td>
<td><strong>0.00115</strong></td>
<td>2.93219</td>
</tr>
<tr>
<td>80</td>
<td>1</td>
<td>0.00493</td>
<td>0.00451</td>
<td><strong>0.00008</strong></td>
</tr>
<tr>
<td>80</td>
<td>2</td>
<td>0.00222</td>
<td>0.00177</td>
<td><strong>0.00009</strong></td>
</tr>
<tr>
<td>80</td>
<td>3</td>
<td>5.91731</td>
<td><strong>0.00007</strong></td>
<td>5.90312</td>
</tr>
</tbody>
</table>

Table 4: Number of cases where IG brings better results than IGR (nxm)

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>40</td>
<td>72</td>
<td>74</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>78</td>
<td>81</td>
<td>26</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>81</td>
<td>81</td>
<td>13</td>
</tr>
<tr>
<td>40</td>
<td>1</td>
<td>81</td>
<td>81</td>
<td>10</td>
</tr>
<tr>
<td>80</td>
<td>1</td>
<td>81</td>
<td>81</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 5: Number of cases where IG brings better results than IGR (nxθs)

<table>
<thead>
<tr>
<th>n</th>
<th>θs</th>
<th>1</th>
<th>10</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>74</td>
<td>54</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>77</td>
<td>55</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>67</td>
<td>54</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>64</td>
<td>54</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>59</td>
<td>54</td>
<td>54</td>
<td></td>
</tr>
</tbody>
</table>

Another analysis regards the influence of the sequence-dependent setup times on the performance of the algorithms. According the results presented in table 7, the IG algorithm seems to achieve better results when θs increases.

The last analysis performed in this paper is about the effects of the variation of the order’s size on the behavior of the algorithm. Unlike the previous analysis, Table 8 shows that, for larger instances, the size order does not induce a different relative performance between both IG-based algorithms.

6. Final Remarks

This paper presented three different approaches to perform an integrated planning of a production system composed by parallel machines and a distribution system represented as a single vehicle with multiple routes.

The mixed-integer model presented in section 2.1 was able to find optimal solution of all small-sized instances. Moreover, it was able to validate the results found by the other solution procedures.
A novel constructive algorithm was then introduced in section 3. More than be able to find a solution for the problem, the analysis performed in the initialization procedures of this algorithm reveals that, in a similar manner that happens with the NEH algorithm, the initial sequence of the input set is closely related to the quality of the solutions found.

Extending the constructive algorithm, section 4 presented two heuristics based on the Iterated Greedy procedure. Although both algorithms could find better results, it was clear that a random destruction based on a single job perform better than removing all jobs from a random route.

Although the results found in this paper were promising, this research can be extended in mainly two streams: the first one, relates to further analysis of the MIP model and advanced solution procedures that could allow one to apply the MIP model on larger instance sizes. The second stream is about developing different ordering procedures to both the constructive algorithm and the IG procedures.

References


Table 8: Number of cases where IG brings better results than IGR (nθσ)

<table>
<thead>
<tr>
<th>n</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>70</td>
<td>60</td>
<td>56</td>
</tr>
<tr>
<td>10</td>
<td>60</td>
<td>64</td>
<td>61</td>
</tr>
<tr>
<td>20</td>
<td>58</td>
<td>57</td>
<td>60</td>
</tr>
<tr>
<td>40</td>
<td>57</td>
<td>58</td>
<td>57</td>
</tr>
<tr>
<td>80</td>
<td>55</td>
<td>55</td>
<td>57</td>
</tr>
</tbody>
</table>

Ng, C. e Lu, L. (2012). On-line integrated production and outbound distribution scheduling to minimize the maximum delivery completion time. *Journal of Scheduling*, 15(3):391 – 398. Arrival time; Asymptotic competitive ratio; Competitive ratio; Completion time; Integrated production; Online algorithms; Processing time; Scheduling problem;


