

Document downloaded from:

<http://hdl.handle.net/10251/44606>

This paper must be cited as:

Hatami, S.; Ruiz García, R.; Andrés Romano, C. (2013). The distributed assembly permutation flowshop scheduling problem. *International Journal of Production Research*. 51(17):5292-5308. doi:10.1080/00207543.2013.807955.



The final publication is available at

<http://dx.doi.org/10.1080/00207543.2013.807955>

Copyright Taylor & Francis: STM, Behavioural Science and Public Health Titles

# The Distributed Assembly Permutation Flowshop Scheduling Problem

Sara Hatami<sup>a</sup>, Rubén Ruiz<sup>b,\*</sup>, Carlos Andrés Romano<sup>c</sup>

<sup>a</sup>*Departamento de Estadística e Investigación Operativa, Facultad de Ciencias Matemáticas, Universitat de València, Spain*

<sup>b</sup>*Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación, Edificio 8G, Acc. B. Universitat Politècnica de València, Camino de Vera s/n, 46021, València, Spain*

<sup>c</sup>*Departamento de Organización de Empresas, Universidad Politècnica de València, Camino de Vera s/n, 46021, València, Spain*

---

## Abstract

Nowadays, improving the management of complex supply chains is key to become competitive in the 21<sup>st</sup> century global market. Supply chains are composed of multi-plant facilities that must be coordinated and synchronized to cut waste and lead times. This paper proposes a Distributed Assembly Permutation Flowshop Scheduling Problem (DAPFSP) with two stages to model and study complex supply chains. This problem is a generalization of the Distributed Permutation Flowshop Scheduling Problem (DPFSP) presented by Naderi and Ruiz (2010). The first stage of the DAPFSP is composed of  $f$  identical production factories. Each one is a flowshop that produces jobs to be assembled into final products in a second assembly stage. The objective is to minimize the makespan. We present first a Mixed Integer Linear Programming model (MILP). Three constructive algorithms are proposed. Finally, a Variable Neighborhood Descent (VND) algorithm has been designed and tested by a comprehensive ANOVA statistical analysis. The results show that the VND algorithm offers good performance to solve this scheduling problem.

*Keywords:* Distributed assembly flowshop, Variable neighborhood descent

---

---

\*Corresponding author

*Email addresses:* [hatami@alumni.uv.es](mailto:hatami@alumni.uv.es) (Sara Hatami), [r Ruiz@eio.upv.es](mailto:r Ruiz@eio.upv.es) (Rubén Ruiz), [candres@omp.upv.es](mailto:candres@omp.upv.es) (Carlos Andrés Romano)

## 1. Introduction

Assembly systems have been widely studied in the last decade given their practical interest and applications. An assembly flowshop is a hybrid production system where various production operations are independently and concurrently performed to make parts that are delivered to an assembly line (Koulamas and Kyparisis, 2001). In assembly systems, a wide variety of final products can be made from a given number of different assembled parts. Assembly programs represent relationships between the different parts which must be assembled from a set of suppliers.

Nowadays a single supplier or production factory is rare. As a matter of fact, production systems with more than one production center (named distributed manufacturing systems) are quite usual as they play an important role in practice (Moon et al., 2002). The benefits of distributed manufacturing systems include achieving higher product quality, lower production costs and fewer management risks (Wang, 1997; Kahn et al., 2004; Chan et al., 2005). From a manager's point of view, scheduling in distributed systems is more complicated than in single-factory scheduling problems. In single-factory problems, the only objective is to find a job schedule for a set of machines, while an important additional decision in the distributed problem is allocating jobs to suitable factories. Therefore, two decisions have to be made; job allocation to factories and job scheduling at each factory. Different job allocations to different factories result in different production schedules, which consequently affects supply chain performance (Chan et al., 2005).

This paper contemplates flowshop scheduling as a production system for each factory or supplier in the distributed problem. The flowshop scheduling problem (FSP) is composed of a set of  $M$  of  $m$  machines where each job of a set  $N$  of  $n$  jobs must be processed in each machine. The number of operations per job is equal to the number of machines. The  $i^{th}$  operation of each job is processed in machine  $i$ . Therefore, one job can start in machine  $i$  only after it has been completed in machine  $i - 1$ , and if machine  $i$  is free. The processing times of each job in the machines are known in advance, non negative and deterministic. In FSPs, a number of assumptions are made (Baker, 1974): all jobs are available for processing at time 0; machines are continuously available (no breakdowns); each machine can process only one job at a time; each job can be processed in only one machine at a time; once the processing of a given job has started in a given machine, it cannot be interrupted and processing continues until completion (no preemption); setup

38 times are sequence-independent and are either included in the processing  
39 times or ignored; infinite in-process storage is allowed.

40 In the FSP, there are  $n!$  possible job permutations for each machine.  
41 Therefore, the total number of solutions for a flowshop problem with  $m$  ma-  
42 chines is  $(n!)^m$ . To simplify the problem, it is assumed that all machines have  
43 the same job permutation. In other words, if one job is at the  $j^{th}$  position on  
44 machine 1, then this job has to be at the  $j^{th}$  position on all other machines as  
45 well. With this simplifying assumption the FSP is referred to as Permutation  
46 Flowshop Scheduling Problem (PFSP) with  $n!$  possible solutions.

47 This paper studies the Distributed Assembly Flowshop Scheduling Prob-  
48 lem (DAPFSP). It is a combination of the DPFSP and the Assembly Flow-  
49 shop Scheduling Problem (AFSP), and consists of two stages: production  
50 and assembly. The first stage consists of a set  $F$  of  $f$  identical factories or  
51 production centers where a set  $N$  of  $n$  jobs have to be scheduled. All factories  
52 are capable of processing all jobs and each factory is a PFSP with a set  $M$   
53 of  $m$  machines. Factories are assumed to be identical. Processing times are  
54 denoted by  $p_{ij}$ ,  $i \in M$ ,  $j \in N$ . The second stage is a single assembly factory  
55 with an assembly machine,  $M_A$ , which assembles jobs by using a defined as-  
56 sembly program to make a set  $T$  of  $t$  different final products. Each product  
57 has a defined assembly program; in other words, each product consists of  
58 some defined jobs.  $N_h$  and  $J_j$  are used, respectively, to represent product  $h$   
59 assembly program and the jobs that belong to the product  $h$  assembly pro-  
60 gram,  $N_h : \{J_j\}, j \in N_h$ . Each product  $h$  has  $|N_h|$  jobs and job  $j$  is needed  
61 for the assembly of one product. Therefore,  $\sum_{h=1}^t |N_h| = n$ . Product  $h$  as-  
62 sembly can start only when all jobs that belong to  $N_h$  have been completed  
63 in the factories. The considered objective is to minimize the makespan at the  
64 assembly factory.

65 The next section presents a short literature review. Section 3 provides  
66 a Mixed Integer Linear Programming (MILP) model to solve the consid-  
67 ered problem. Section 4 introduces three constructive heuristics, while Sec-  
68 tion 5 presents an iterative method based on Variable Neighborhood Descent  
69 (VND) to improve results further. Section 6 describes a complete compu-  
70 tational evaluation of the MILP model and proposed algorithms, where the  
71 performance of the proposed approaches is discussed in order to assess the in-  
72 fluence of the number of jobs, machines, factories, products and some solver  
73 options on the results. Finally, Section 7 offers conclusions, remarks and  
74 venues for future research.

## 75 2. Literature review

76 The DPFSP can be viewed as a generalized version of the PFSP. This  
77 problem is one of the most researched topics in the scheduling literature  
78 (Pinedo, 2012; Dong et al., 2009; Zobolas et al., 2009; Laha and Sarin, 2009;  
79 Vallada and Ruiz, 2010; Xu et al., 2011; Zhang and Li, 2011; Chen et al.,  
80 2012; Pan and Ruiz, 2012).

81 In the PFSP, more attention has been paid to makespan minimization.  
82 The practical implication is obvious: minimizing the makespan leads to the  
83 minimization of the total production run (Framinan et al., 2002). There are  
84 some proposed effective rules and algorithms for the PFSP (Johnson, 1954;  
85 Nawaz et al., 1983). A comprehensive review and evaluation has been made  
86 by Ruiz and Maroto (2005), Vallada et al. (2008) and Pan and Ruiz (2013).

87 Regarding the assembly scheduling problem, Lee et al. (1993) presented  
88 a three-machine assembly-type flowshop scheduling problem by considering  
89 makespan minimization as the objective function. In their considered model,  
90 each product is composed of two types of jobs, where type  $a$  and  $b$  are pro-  
91 cessed by machine  $M_a$  and  $M_b$ , respectively, and machine  $M_2$  assembles the  
92 two jobs into a product. These authors also present a branch-and-bound so-  
93 lution scheme and an approximate solution procedure. Later, Potts et al.  
94 (1995) extended the model of Lee et al. (1993) by considering  $m$  parallel  
95 production machines instead of the first two production machines. They ap-  
96 ply the compact vector summation technique to find approximated solutions  
97 with worse-case absolute performance guarantees. Hariri and Potts (1997)  
98 developed a branch-and-bound algorithm for the same model as Potts et al.  
99 (1995). Moreover, Tozkapan et al. (2003) considered a two-stage assembly  
100 scheduling problem by minimizing the total weighted flow time as an objec-  
101 tive function. They developed a lower bound and a dominance criterion, and  
102 incorporated them into a branch-and-bound procedure. They also presented  
103 a heuristic procedure to find an initial upper bound. Al-Anzi and Allahverdi  
104 (2006) addressed the model presented by Tozkapan et al. (2003) and mini-  
105 mized the total completion time of all the jobs. They used metaheuristics to  
106 solve their model and proposed simulated annealing (SA), tabu search (TS),  
107 and hybrid tabu search heuristics for general cases.

108 Despite the innumerable literature related to PFSP and AFSP, there are  
109 few studies about the distributed problems. Jia et al. (2002) reported a web-  
110 based system to enable production scheduling (a job shop problem) for the  
111 distributed manufacturing environment and a Genetic Algorithm (GA) was

112 adopted to solve the problem. Jia et al. (2003) presented a modified GA to  
113 deal with distributed job shop scheduling problems. Later, Jia et al. (2007)  
114 proposed a new approach to determine good combinations of factories to  
115 manufacture jobs. An adaptive GA for distributed scheduling problems was  
116 proposed by Chan et al. (2005). The same authors proposed a GA with  
117 dominant genes for solving distributed scheduling problems in an FMS envi-  
118 ronment in Chan et al. (2006a). Furthermore, Chan et al. (2006b) proposed  
119 a GA to deal with distributed flexible manufacturing system (FMS) sub-  
120 ject to machine maintenance constraints. Naderi and Ruiz (2010) introduced  
121 the DPFSP for the first time. They developed six different MILPs for the  
122 considered problem and proposed two simple factory assignment rules and  
123 14 heuristics based on dispatching rules, effective constructive heuristics and  
124 VND methods. Liu and Gao (2010) proposed an electromagnetism-like mech-  
125 anism (EM) algorithm for the same problem. The same authors, in Gao and  
126 Chen (2011a) proposed a GA-based algorithm, denoted by GA-LS, Gao and  
127 Chen (2011b) a constructive heuristic algorithm enhanced with a dispatching  
128 rule, Gao et al. (2012b) a knowledge-based genetic algorithm and Gao et al.  
129 (2012a) a Variable Neighborhood Descent (VND) algorithm.

130 To the best of our knowledge, no further literature exists on DAPFSP,  
131 so this is the first effort that considers the assembly flowshop problem in a  
132 distributed manufacturing setting.

### 133 **3. Mixed Integer Linear Programming model**

134 A mathematical model is an abstract and good approach that uses math-  
135 ematical language to describe in detail a problem. There are many papers  
136 related to the flowshop problem which use MILP modeling; for example, we  
137 can cite Stafford et al. (2005); Tseng and Stafford (2008); Ching-Jong and  
138 Li-Man (2008) and Naderi and Ruiz (2010), to name just a few.

139 We first define the model indexes, parameters and variables in Table 1, and  
140 present the MILP afterwards. The proposed MILP model is inspired by the  
141 fifth mathematical model that is presented in Naderi and Ruiz (2010) for the  
142 DPFSP that was shown to outperform the other models tested in that paper.

143 The objective function of the model is to minimize a makespan:

$$144 \text{Min } C_{\max}$$

| Index      | Description  |
|------------|--|
| $k, j$     | denotes jobs, $k, j = 0, 1, \dots, n$ , where 0 presents a dummy job                 |
| $i$        | denotes machines at each factory, $i = 1, \dots, m$                                  |
| $l, s$     | denotes products, $l, s = 0, 1, \dots, t$ , where 0 presents a dummy product         |
| $M$        | A sufficiently large positive number   |
| Parameters | Description  |
| $n$        | number of jobs   |
| $m$        | number of machines   |
| $f$        | number of factories  |
| $t$        | number of products   |
| $p_{ij}$   | processing time of job $j$ on machine $i$  |
| $pp_s$     | processing time of product $s$ at the assembly stage                                 |
| $G_{js}$   | Binary parameter equal to 1 if job $j$ belongs to product $s$ , and 0 otherwise      |
| Variable   | Description  |
| $X_{kj}$   | binary variable equal to 1 if job $k$ is an immediate predecessor of job $j$         |
| $Y_{ls}$   | binary variable equal to 1 if product $l$ is an immediate predecessor of product $s$ |
| $C_{ij}$   | completion time of job $j$ on machine $i$  |
| $CA_s$     | completion time of product $s$ on assembly stage                                     |
| $C_{\max}$ | makespan   |

Table 1: indexes, parameters and variables used in MILP mathematical model.

145 and the constraints of the model are:

$$\sum_{k=0, k \neq j}^n X_{kj} = 1 \quad \forall j \quad (1)$$

$$\sum_{j=0, k \neq j}^n X_{kj} \leq 1 \quad \forall k \quad (2)$$

$$\sum_{j=1}^n X_{0j} = f \quad (3)$$

$$\sum_{k=1}^n X_{k0} = f - 1 \quad (4)$$

$$X_{kj} + X_{jk} \leq 1 \quad \forall j \in \{1, \dots, n-1\}, j > k \quad (5)$$

$$C_{ij} \geq C_{i-1j} + p_{ij} \quad \forall i, j \quad (6)$$

$$C_{ij} \geq C_{ik} + p_{ij} + (X_{kj} - 1) \cdot M \quad \forall k, j \neq k, i \quad (7)$$

$$\sum_{l=0, l \neq s}^t Y_{ls} = 1 \quad \forall s \quad (8)$$

$$\sum_{s=1, l \neq s}^t Y_{ls} \leq 1 \quad \forall l \quad (9)$$

$$Y_{ls} + Y_{sl} \leq 1 \quad \forall l \in \{1, \dots, t-1\}, s > l \quad (10)$$

$$CA_s \geq (C_{mj} \cdot G_{js}) + pp_s \quad \forall j, s \quad (11)$$

$$CA_s \geq CA_l + pp_s + (Y_{ls} - 1) \cdot M \quad \forall l, s \quad (12)$$

$$C_{\max} \geq CA_s \quad \forall s \quad (13)$$

$$X_{kj} \in \{0, 1\} \quad \forall k, j, k \neq j \quad (14)$$

$$Y_{ls} \in \{0, 1\} \quad \forall l, s, l \neq s \quad (15)$$

$$C_{ij} \geq 0 \quad \forall i, j \quad (16)$$

$$CA_s \geq 0 \quad \forall s \quad (17)$$

147 Note that  $C_{0j} = CA_0 = 0, \forall j$ . Constraint set (1) controls and ensures  
 148 that each job must have exactly one predecessor. Constraint set (2) indicates  
 149 that each job has one succeeding job at the most. Constraint set (3) enforces  
 150 that dummy job 0 has to have  $f$  predecessor in the final sequence. Constraint  
 151 set (4) also enforces that dummy job 0 must be a successor  $f - 1$  times (there  
 152 is no dummy job at the end of the sequence). Constraint set (5) controls and  
 153 ensures that a job cannot be both a predecessor and successor of another  
 154 job at the same time. Constraint set (6) enforces the processing of job  $j$  in  
 155 machine  $i$  when the processing at machine  $i - 1$  is completed. Constraint set  
 156 (7) determines that if job  $j$  is placed immediately after job  $k$ , its processing  
 157 at machine  $i$  cannot start before the processing of job  $k$  in machine  $i$  finishes.  
 158 Constraints (8) and (9) force that each product should have one predecessor  
 159 and at most one succeeding product in the assembly factory, respectively,  
 160 constraint (10) controls that a product cannot be both a predecessor and a  
 161 successor of another product at the same time. Constraint (11) implies that  
 162 each product  $h$  cannot begin its assembly before all the jobs in its assembly  
 163 program are completed in the last machine  $m$ . Constraint set (12) determines  
 164 that if product  $s$  is placed immediately after product  $l$ , its processing on  
 165 assembly machine cannot start before the processing of product  $l$  in assembly  
 166 machine finishes. Constraint (13) defines the makespan, while constraints  
 167 (14)-(17) define the domain of the decision variables.

168 The significant point of this model is that there is no index for facto-  
 169 ries. Sequence-based variables are hence used with a set of  $f$  dummy jobs.



170 These dummy jobs divide all the jobs into subsequences and assign them  
 171 to each factory (i.e., all jobs placed between the first dummy job and the  
 172 second dummy job belong to the first factory, and so on). For example,  
 173 if one of the possible solutions for a problem with  $n = 8$  and  $f = 3$  is  
 174  $X_{0,2} = X_{2,3} = X_{3,5} = X_{5,0} = X_{0,6} = X_{6,1} = X_{1,4} = X_{4,0} = X_{0,7} = X_{7,8} = 1$ ,  
 175 then the sequence is  $\{0, 2, 3, 5, 0, 6, 1, 4, 0, 7, 8\}$ , where partial job sequences  
 176  $\{2, 3, 5\}$ ,  $\{6, 1, 4\}$  and  $\{7, 8\}$  are assigned to factories 1, 2 and 3, respectively.

#### 177 4. Heuristic methods

178 As mentioned in the paper of Naderi and Ruiz (2010), the DPFSP is an  
 179 NP-Complete problem (if  $n > f$ ); accordingly, the DAPFSP with an addi-  
 180 tional assembly stage as a further stage is certainly a NP-Complete problem.  
 181 Therefore, it is necessary to develop a heuristic approach to solve large-sized  
 182 problems. In order to solve instances of realistic size in this problem, three  
 183 constructive simple heuristics are proposed.

184 For the assignment of jobs to factories, the two rules, of Naderi and Ruiz  
 185 (2010) are used.

- 186 1. Assign job  $j$  to the factory which has the lowest current  $C_{\max}$ , (NR<sub>1</sub>).
- 187 2. Assign job  $j$  to the factory which has the lowest  $C_{\max}$  after including  
 188 job  $j$ , (NR<sub>2</sub>).

189 Using these two factory allocation rules, three heuristics are presented to  
 190 schedule jobs.

191

##### 192 4.1. Heuristic 1

193 We first introduce some necessary notation. An example with  $n = 9$ ,  
 194  $m = 2$ ,  $f = 2$  and  $t = 3$ , this is, 9 jobs, 2 factories with a flowshop of  
 195 two machines each and three products to assemble, is employed to explain  
 196 expressions and heuristics in detail. Table 2 shows the processing times of  
 197 the jobs and assembly processing times of products. The products' assembly  
 198 programs are:  $N_1 = \{3, 4, 6\}$ ,  $N_2 = \{1, 2, 8, 9\}$  and  $N_3 = \{5, 7\}$ .  $\pi$  represents  
 199 a *product sequence*, e.g.,  $\pi : \{1, 3, 2\}$  is a possible product sequence for the  
 200 given example. As mentioned before, each product  $h$  is made up of  $|N_h|$   
 201 jobs and  $\pi_h$  is the *partial job sequence* of product  $h$ , e.g.,  $\pi_1 : \{6, 4, 3\}$ ,  
 202  $\pi_2 : \{1, 9, 8, 2\}$  and  $\pi_3 : \{7, 5\}$ . A *complete job sequence*,  $\pi_T$ , is constructed

203 by putting together all partial job sequences, following the product sequence  
204  $\pi$ , e.g.,  $\pi_T : \{6, 4, 3, 7, 5, 1, 9, 8, 2\}$ .

205 The shortest processing time (SPT) is a well-known dispatching rule for  
206 the PFSP. In the SPT, the job with the shortest processing time is processed  
207 first. This rule tends to reduce the work-in-process inventory, the average  
208 throughput time, and average job lateness (Vollmann et al., 2005). Hence  
209 the SPT is used to determine the product sequence in the assembly machine.

210 Heuristic 1 begins by applying the SPT rule for the assembly operation  
211 times to obtain  $\pi$ . A heuristic which is based on Framinan and Leisten (2003)  
212 heuristic (FL) is applied on the jobs that belong to a given product, to  
213 obtain a good partial job sequence for each product. The heuristic evaluates  
214 the completion times of the jobs that belong to product  $h$ . Set  $R_h$  is made  
215 by sorting jobs in ascending order of completion times. The first two jobs  
216 of  $R_h$  are selected and inserted into  $S_h$ . When there are only two jobs in  
217  $S_h$ , all pairwise exchanges are checked and  $S_h$  is updated with the one that  
218 results in the best makespan. The next step is removing the third job in  $R_h$   
219 and inserting it in all possible positions of  $S_h$ . The sequence with the best  
220 makespan will be selected. All possible sequences by carrying out pairwise  
221 exchanges between jobs are evaluated again. The process continues until all  
222 jobs have been considered.  $S_h$  is the partial job sequence for product  $h$ , ( $\pi_h$ ).  
223  $\pi_T$  is constructed by putting together all  $\pi_h$  and jobs are assigned to factories  
224 from  $\pi_T$  by using  $NR_1$  or  $NR_2$ , which respectively result in the  $H_{11}$  or  $H_{12}$   
225 heuristics.

226 Pseudocode 1 explains heuristics  $H_{11}$  and  $H_{12}$  in detail:

---

**Pseudocode 1** Outline of the  $H_{11}/H_{12}$  heuristic.

---

- Obtain product sequence  $\pi$  after applying the SPT rule on product assembly processing times,  $\pi = \{\pi(1), \pi(2), \dots, \pi(t)\}$ ; ( $\pi(1)$ : The first product in product sequence)
- Determine partial job sequence for all products using the proposed algorithm based on FL heuristic ( $\pi_h$ : partial job sequence for product  $h$ )
- Construct complete job sequence ( $\pi_T$ ) by putting together all partial job sequences ( $\pi_h$ ), following the product sequence,  $\pi$
- Assign all jobs in  $\pi_T$  to factories using  $NR_1$  to make  $H_{11}$  and using  $NR_2$  to make  $H_{12}$

227 Let us now apply proposed heuristics to the example.  $\pi : \{1, 3, 2\}$  is  
228 the product sequence obtained after applying the SPT rule to the assembly  
229 processing times of the products. The next step is to find a good partial job  
230 sequence for each product. As mentioned before, each product has a defined  
231 assembly program that includes a defined set of  $|N_h|$  jobs. Completion time

|          |   | Jobs      |   |    |           |   |    |           |   |  |
|----------|---|-----------|---|----|-----------|---|----|-----------|---|--|
| Machines | 1 | 2         | 3 | 4  | 5         | 6 | 7  | 8         | 9 |  |
| $M_1$    | 1 | 5         | 7 | 9  | 9         | 3 | 8  | 4         | 2 |  |
| $M_2$    | 3 | 8         | 5 | 7  | 3         | 4 | 1  | 3         | 5 |  |
|          |   | Product 1 |   |    | Product 2 |   |    | Product 3 |   |  |
| $M_A$    | 6 |           |   | 19 |           |   | 12 |           |   |  |

Table 2: Processing times of the jobs and assembly processing times of the products for the example.

232 for each job at the production stage is the summation of each job processing  
 233 times on all machines,  $\sum_{i=1}^m p_{ij}$ . Therefore, completion times for set of jobs  
 234 of the product 1,  $N_1 = \{3, 4, 6\}$  are  $C_{23} = 12$ ,  $C_{24} = 16$ ,  $C_{26} = 7$ . Set  
 235  $R_1$  is obtained by arranging jobs in an increasing completion time order;  
 236  $R_1 = \{6, 3, 4\}$ . The first two jobs of  $R_1$  are selected and included into  $S_1$ .  
 237 All possible sequences resulting from pairwise exchanges of the first two jobs  
 238 in  $S_1$  are calculated:  $\{6, 3\}$  and  $\{3, 6\}$  which result in makespans values  
 239 of 15 and 16, respectively. The sequence with the minimum makespan is  
 240  $S_1 : \{6, 3\}$ . The third job in  $R_1$ , (4) is inserted into all possible positions of  
 241  $S_1$ . The obtained partial job sequences are:  $\{4, 6, 3\}$ ,  $\{6, 4, 3\}$  and  $\{6, 3, 4\}$   
 242 and their makespans in the production stage are: 25, 24, 26, respectively.  
 243 As a result, the second is the best position for job 4 and  $S_1$  is updated  
 244 to  $\{6, 4, 3\}$ . In the next step, general pairwise exchanges are carried out on  
 245 the updated  $S_1$ ; hence, the partial job sequences are:  $\{4, 6, 3\}$ ,  $\{6, 3, 4\}$  and  
 246  $\{3, 4, 6\}$  and, subsequently, their makespans in the production stage are, 25,  
 247 26, 27, respectively. If a better makespan is obtained, then  $S_1$  is updated.  
 248 This process continues until all jobs have been inserted into  $S_1$ .  $\pi_1$  is the  
 249 final updated  $S_1$ , which is equal to  $\{6, 4, 3\}$ . By following the same method,  
 250 the partial job sequences for the other products are:  $\pi_2 = \{1, 9, 8, 2\}$  and  
 251  $\pi_3 = \{5, 7\}$  with partial makespans of 20 and 18, respectively. Hence  $\pi_T$   
 252 is  $\{6, 4, 3, 5, 7, 1, 9, 8, 2\}$ . The final step is to assign jobs in  $\pi_T$  to factories  
 253 by using  $NR_1/NR_2$  to obtain  $H_{11}/H_{12}$ .  $C_{\max}$  of  $H_{11}$  and  $H_{12}$  are 55 and 53,  
 254 respectively. The Gantt chart of the considered example after applying  $H_{11}$   
 255 is shown in Figure 1.

#### 256 4.2. Heuristic 2

257 The idea of the second heuristic is to give priority to products whose jobs  
 258 are completed in the production stage sooner. This concept is noted as the

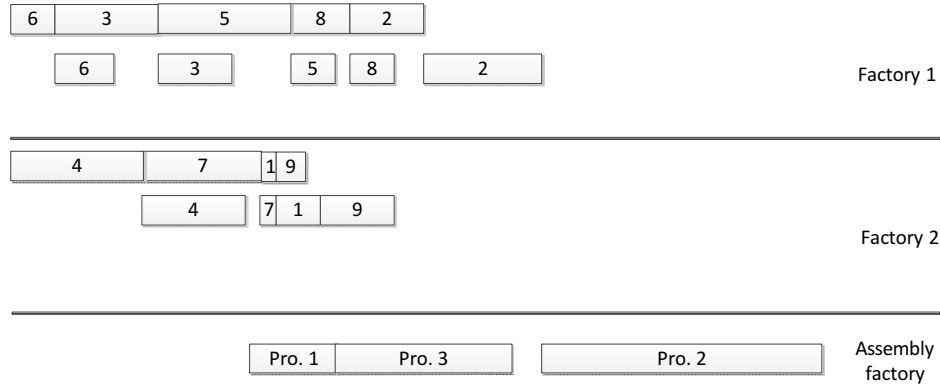


Figure 1: Gantt chart of  $H_{11}$  for the example.

259 earliest start time to assemble product  $h$ ,  $E_h$ . The procedure that is used  
 260 in  $H_{11}$  and  $H_{12}$  to find partial job sequences of products ( $\pi_h$ ) also is used  
 261 in heuristic 2.  $E_h$ , is calculated by using  $NR_1$  or  $NR_2$  to assign jobs in each  
 262 partial job sequence to factories.  $\pi$  is built by sorting  $E_h$  in ascending order.  
 263 A detailed explanation is shown in Pseudocode 2.

**Pseudocode 2** Outline of the  $H_{21}/H_{22}$  heuristic.

- Determine partial job sequences of products using proposed algorithm based on FL heuristic ( $\pi_h$ : partial job sequence for product  $h$ )
- Calculate the earliest start time to assemble each product  $h$ ,  $E_h$ , using  $NR_1$  and  $NR_2$  to assign jobs of the partial job sequences, respectively for  $H_{21}$  and  $H_{22}$
- Sort  $E_h$  in ascending order for all the products to obtain product sequence,  $\pi$ :  $\{\pi(1), \pi(2), \dots, \pi(t)\}$
- Construct complete job sequence ( $\pi_T$ ) by putting together all partial job sequences ( $\pi_h$ ), following the product sequence,  $\pi$
- Assign all jobs of  $\pi_T$  to factories using  $NR_1$  to make  $H_{21}$  and using  $NR_2$  to make  $H_{22}$

264 The last example data is also used to clarify the second proposed heuristic.  
 265  $E_h$  is calculated by applying job assignment rules ( $NR_1$  for the  $H_{21}$  and  
 266  $NR_2$  for the  $H_{22}$ ) for the partial job sequence of product  $h$ . Therefore, the  
 267 earliest start times for assembling products by considering  $NR_2$  are  $E_1 = 15$ ,  
 268  $E_2 = 15$  and  $E_3 = 12$ . The product sequence  $\pi$  is obtained by sorting  $E_h$  in  
 269 ascending order,  $\pi$  :  $\{3, 2, 1\}$ . As a result, the complete job sequence,  $\pi_T$ , will  
 270 be:  $\{5, 7, 1, 9, 8, 2, 6, 4, 3\}$ . The final results of  $C_{\max}$  for  $H_{21}$  and  $H_{22}$  are equal  
 271 to 51 and 50, respectively.

272 *4.3. Heuristic 3*

273 The third proposed heuristic is similar to the second one. The difference  
 274 is in the construction of the partial job sequences of each product ( $\pi_h$ ). While  
 275 heuristic 2 uses a heuristic based on FL, heuristic 3 employs the more simple  
 276 SPT rule. Our intention is to test if a simpler constructive heuristic gives  
 277 similar results.

278 Table 3 shows the  $C_{mj}$  of the jobs, the partial job sequence for each  
 279 product, after applying the SPT rule and  $E_h$  of product  $h$  in the columns for  
 280 the example.

| Product ( $h$ ) | Job ( $j$ ) | $C_{2j}$ | $\pi_h$    | $E_h$ |
|-----------------|-------------|----------|------------|-------|
| 1               | 3           | 12       | 6, 3, 4    | 19    |
|                 | 4           | 16       |            |       |
|                 | 6           | 7        |            |       |
| 2               | 1           | 4        | 1, 9, 8, 2 | 15    |
|                 | 2           | 13       |            |       |
|                 | 8           | 7        |            |       |
|                 | 9           | 7        |            |       |
| 3               | 5           | 12       | 7, 5       | 12    |
|                 | 7           | 9        |            |       |

Table 3: Job completion times on the last machine of production stage, products partial job sequence and earliest start time for assembling each product for the example.

281 Product sequence  $\pi$  is  $\{3, 2, 1\}$  after sorting  $E_h$  in ascending order. The  
 282 complete sequence  $\pi_T$  after putting together the partial jobs sequences of  
 283 each product is:  $\{7, 5, 1, 9, 8, 2, 6, 3, 4\}$ . After applying  $NR_1$  to this sequence  
 284 we obtain a  $C_{\max}$  of 51. The  $C_{\max}$  for  $NR_2$  is 50.

285 **5. Variable Neighborhood Descent (VND)**

286 We now present a Variable Neighborhood Descent (VND) method (Hansen  
 287 and Mladenovic, 2001). VND is an enhanced local improvement strategy  
 288 based on the systematic exploration of different neighborhood structures  
 289  $N_1, \dots, N_q$ . A VND starts with the first structure  $N_1$  by performing a local  
 290 search until no further improvements are possible. From this local optimum,  
 291 it continues the local search with neighborhood structure  $N_2$ . If an improved  
 292 solution is found with this structure, the VND goes back to  $N_1$ ; otherwise, it  
 293 continues with  $N_3$ , and so forth. If the last structure  $N_q$  has been applied and  
 294 no further improvements are possible, the solution represents a local optimum  
 295 with respect to all neighborhood structures and the VND terminates.

296 *5.1. Solution representation and VND initialization*

297 In order to represent a solution, a complete sequence of all jobs  $\pi_T$  is  
298 considered, like in the PFSP. We limit the representation so that all jobs from  
299 a product are never separated. The jobs in the complete sequence are assigned  
300 to factories using  $NR_1$  or  $NR_2$ . An example of a solution representation can  
301 be:  $\{6, 4, 3, 1, 9, 8, 2, 5, 7\}$  which is equal to product sequence of  $\{1, 2, 3\}$   
302 with respect to the last example.

303 The VND approach needs an initial solution. Although a random solution  
304 can be used as an initial solution, it is better to use heuristics (Naderi and  
305 Ruiz, 2010; Vallada and Ruiz, 2010; Ruiz and Stützle, 2007). Our approach  
306 uses the six proposed constructive heuristics to obtain the initial solution.  
307 Later we will test six VND versions, each one starting from the result of each  
308 heuristic.

309 *5.2. Neighborhoods and acceptance criterion*

310 Our proposed VND heuristic employs two neighborhood structures, and  
311 both are applied to the complete sequence  $\pi_T$ .

312 The first is referred to as  $LS_P$  and is a product local search. It attempts  
313 to improve the objective function by examining different product sequences.  
314  $LS_P$  works as follows: 1) It provides a list of product sequences by removing  
315 a single product from  $\pi$  and inserting it in all the possible  $t - 1$  positions of  
316 current  $\pi$ ; 2) It evaluates the list of obtained product sequences by converting  
317 them into  $\pi_T$  and assigning the jobs of  $\pi_T$  to factories via  $NR_1$  or  $NR_2$ ; 3) If  
318 one of the obtained  $\pi$  in the list has a better  $C_{\max}$ , then  $\pi$  is updated to the  
319 better product sequence and all the products are reinserted again (a local  
320 search until a local optimum), otherwise the search continues with the next  
321 product.

322 The second neighborhood is  $LS_J$ , tries to find different partial job se-  
323 quences for each product to improve the objective function.  $LS_J$  works as  
324 follows: 1)  $LS_J$  starts with the first product  $h$ , then the local search starts  
325 by removing the first job of  $\pi_h$  and inserting it in all the possible  $|N_h| - 1$   
326 positions of  $\pi_h$ ; 2) Evaluate  $\pi_T$  with all the newly obtained partial job se-  
327 quences for product  $h$ ; 3) If a better objective function is obtained, then  $\pi_h$   
328 is updated and all jobs in  $\pi_h$  are reinserted again until a local optimum is  
329 found. Otherwise, the search continues with the next job in  $\pi_h$ ; 4)  $LS_J$  will  
330 continue with the next product until all products have been considered.

331 Pseudocodes 3 and 4 show the product and the job local search, respec-  
332 tively.

---

**Pseudocode 3** Product Local Search,  $LS_P$ .

---

```
l = 1
while l ≤ t do
  - Remove product a which is placed in position l of π
  - Insert a into all t - 1 possible positions of π
  - Evaluate all obtained π by converting them into πT
  if a better Cmax is obtained then
    - update π
  else
    l = l + 1
  end if
end while
```

---

---

**Pseudocode 4** Job Local Search,  $LS_J$ .

---

```
h = 1
while h ≤ t do
  j = 1
  while j ≤ Nh do
    - Remove job b which is placed at position j of πh
    - Insert b into all |Nh| - 1 possible positions of current πh
    - using the new πh, convert it to πT
    if a better Cmax is obtained then
      - Select the partial job sequence with the best result as the new πh
    else
      j = j + 1
    end if
  end while
  h = h + 1
end while
```

---

**333 6. Computational evaluation**

334 Two complete sets of instances have been generated to test the MILP  
335 model and the proposed heuristics. Due to the complexity of the problem,  
336 and given the number of different characteristics considered, four instance  
337 factors and three test factors are combined at the levels provided in Table 4  
338 for small instances. The test factors are: two commercial solver packages  
339 (*Solver*) are used as solving tools, the number of CPU threads (*Thread*),  
340 where we have tested 1 thread (serial computing) and 2 threads (parallel  
341 computing) and a time limitation *TimeLimit* for the stopping criterion. The  
342 heuristics are also tested in a set of larger instances, which differ in the factors  
343 as listed in Table 5.

344 Processing times in the production stage are fixed to  $U[1, 99]$  as it is usual  
345 in the scheduling literature. The assembly processing times depend on the  
346 number of jobs assigned to each product  $h$  as  $U[1 \times |N_h|, 99 \times |N_h|]$ . The total

| Instance factor     | Symbol           | Number of levels | Values                                       |
|---------------------|------------------|------------------|--|
| Number of jobs      | $n$              | 5                | 8, 12, 16, 20, 24                            |
| Number of machines  | $m$              | 4                | 2, 3, 4, 5                                   |
| Number of factories | $f$              | 3                | 2, 3, 4                                      |
| Number of products  | $t$              | 3                | 2, 3, 4                                      |
| Test factor         | Symbol           | Number of levels | Values                                       |
| Solver              | <i>Solver</i>    | 2                | CPLEX 12.3, GUROBI 4.6.1                     |
| Thread              | <i>Thread</i>    | 2                | Serial computing (1), Parallel computing (2) |
| Time limitation     | <i>TimeLimit</i> | 2                | 900s, 3600s                                  |

Table 4: Instance and test factors for the small instances.

| Instance factor     | Symbol | Number of levels | Values        |
|---------------------|--------|------------------|---------------|
| Number of jobs      | $n$    | 3                | 100, 200, 500 |
| Number of machines  | $m$    | 3                | 5, 10, 20     |
| Number of factories | $f$    | 3                | 4, 6, 8       |
| Number of products  | $t$    | 3                | 30, 40, 50    |

Table 5: Instance factors for the large instances.

347 number of combinations in the small and large instances are  $5 \times 4 \times 3^2 = 180$   
348 and  $3^4 = 81$ , respectively. There are five replications per combination for  
349 small instances and ten replications for every large combination. Therefore,  
350 the total number of instances is 900 and 810, respectively. All the instances  
351 are available at <http://soa.iti.es>.

352

### 353 6.1. MILP model evaluation

354 A linear programming model has been constructed for each small instance.  
355 It is solved with all the combinations of the test factors, using CPLEX 12.3  
356 and GUROBI 4.6.1 solvers, serial and parallel computing and two time limits  
357 (900s and 3600s). All the tests are carried out in a high performance comput-  
358 ing cluster with 30 blades, each one containing 16 GBytes of RAM memory  
359 and two Intel XEON E5420 processors running at 2.5 GHz. Note that each  
360 processor has 4 physical computing cores (8 per blade). The 30 blade servers  
361 are used only to divide the workload and experimentations. Experiments are  
362 carried out in virtualized Windows XP machines, each with one virtualized  
363 processor with two cores and 2 GB of RAM memory.

364 A categorical variable named “response type” with two values, 0 and 1,  
365 is reported. Value 0 means that an optimum solution is found in the given  
366 time with  $C_{\max}$  value as a result, and 1 means that in 900s or 3600s, a



367 feasible integer solution is found and reported, but it has not been proven  
 368 to be optimal. Moreover, the gap between this solution and the best MILP  
 369 bound is also reported. In the CPU time allowed, the LP model with all  
 370 900 small instances is able to find 516 optimum solutions (57.33 %). Table 6  
 371 summarizes the results, which are categorized by factors of solver, threads and  
 372 time limit. The comparison criteria are: the percentage of optimum solutions  
 373 found ( $\%opt$ ), the average gap as a percentage for the cases in which the  
 374 optimum solution is not found ( $GAP\%$ ) and the average time required in  
 375 seconds. Later we will carry out statistical testing to ascertain the significance  
 376 of the observed differences.

377 It is clear that GUROBI is able to find more optimal solutions than  
 378 CPLEX, and its average gap and average CPU time consumption are smaller  
 379 than CPLEX. Overall, time limit of 3600 seconds and parallel computing  
 380 (2 threads) results in a larger number of optimal solutions, in comparison  
 381 with time limit of 900 seconds and serial computing (1 thread). CPLEX with  
 382 parallel computing (2 threads) results in a greater average gap in comparison  
 383 with serial computing, but this trend is reversed with GUROBI. Among all  
 384 the eight combinations of test factors, GUROBI with two threads and 3600  
 385 seconds time limitation finds more optimum solutions than the others.

| Solver | Time Limit  | 900s   |        | 3600s   |         |
|--------|-------------|--------|--------|---------|---------|
|        | Thread      | 1      | 2      | 1       | 2       |
| CPLEX  | % opt       | 59.44  | 61.22  | 63.11   | 61.89   |
|        | GAP%        | 29.62  | 30.77  | 32.23   | 36.46   |
|        | Av Time (s) | 390.41 | 380.69 | 1426.53 | 1441.80 |
| GUROBI | % opt       | 66.89  | 68.33  | 70.78   | 73.00   |
|        | GAP%        | 2.19   | 2.04   | 1.81    | 1.70    |
|        | Av Time (s) | 328.15 | 315.57 | 1152.36 | 1089.00 |

Table 6: Performance results for solvers, threads and time limit for the small instances.

386 Automatic Interaction Detection (AID) is an advanced statistical tech-  
 387 nique for multivariate analysis, which was developed by Morgan and Sonquist  
 388 (1963). It seeks to find explanatory variables and combinations of these vari-  
 389 ables which are important for lowering variance in the dependent variables.  
 390 AID is a stepwise procedure that subdivides experimental data according  
 391 to one factor through a series of dichotomous splits into a number of mu-  
 392 tually exclusive subgroups. The initial AID was improved by Kass (1980)

393 by including statistical significance testing in the partition process and by  
394 allowing multi-way splits of data resulting in the so-called Chi-squared Au-  
395 tomatic Interaction Detection (CHAID). A modification to the basic CHAID  
396 algorithm, called an exhaustive CHAID, introduced by Biggs et al. (1991),  
397 performs a more thorough merging and testing of factor variables.

398 An exhaustive CHAID is used to draw a decision tree to analyze the ef-  
399 fect and interactions of the factors for the averages observed in Table 6. AID  
400 techniques are used in different areas like market research, psychology, edu-  
401 cation, scheduling, etc. Recently, CHAID was employed by Ruiz et al. (2008)  
402 to analyze a complex non distributed scheduling problem MILP model. Also,  
403 Ruiz and Andrés-Romano (2011) employed CHAID to analyse a MILP in a  
404 problem with unrelated parallel machines with resource-assignable sequence-  
405 dependent setup times. Naderi and Ruiz (2010) also used CHAID to analyze  
406 several models for the distributed permutation flowshop scheduling problem.

407 The exhaustive CHAID method is used to analyze the MILP results,  
408 which were previously presented. The factors, either serial computing or par-  
409 allel computing (*Threads*), *solver*, *n*, *m*, *f* and *t*, are controlled. We introduce  
410 all the data of both stopping CPU time criteria so the factor time is controlled  
411 as well. The response variable is the type of solution reached by CPLEX and  
412 GUROBI with two possible values (0 and 1). We use the PASW statistics  
413 version 18 software and set a high confidence level for splitting of 99.9%, as  
414 well as a Bonferroni adjustment for the multi-way splits, which compensates  
415 the statistical bias in multi-way paired tests.

416 In Figure 2, the root node contains the total percentage of the cases were  
417 instances were solved optimally (type 0) and the total number of cases. The  
418 most significant factor is the number of jobs or *n*, and the next level is divided  
419 into one node for each possible *n* value. The *p*-value obtained for this split  
420 comes very close to 0 and the result of the  $\chi^2$  statistic is very high, meaning  
421 that the split is done with a very high level of confidence; i.e., *n* is the most  
422 influential factor on the response variable with a very statistically significant  
423 effect.

424 Among the resulting five nodes, as the *n* value increases, the number of  
425 cases for which an optimal solution is found decreases. As a matter of fact, for  
426 *n* = 20 and 24, only 35.6% of the instances are optimally solved. After this  
427 first multi-way split, nodes are split into the number of factories factor, except  
428 for *n* = 8. It is logical that when there is a larger amount of factories, jobs have  
429 more options for allocations, and the completion time of jobs also shortens.  
430 Hence, the earliest possible time to start product assembly also shortens, and

431 the possibility of finding a better solution increases. The number of products  
432  $t$  is the third next important factor, except for node  $n = 12 / f = 3$ , where  
433 number of machines is a significant factor. No further statistically significant  
434 divisions are found and the stopping criterion for branching is met for nodes  
435  $n = 12 / f = 4$  and  $n = 24 / f = 2$ . The number of products factor shows  
436 the same trend as the second important factor (number of factories); that  
437 is, a higher percentage of optimal solutions is found when there is a larger  
438 number of products. If the number of jobs is constant and the number of  
439 products increases, fewer jobs will be dedicated to each product on average,  
440 so finding a better partial job sequence for each product is easier.

441 As seen, apart from a few isolated cases, the effect of type of solver, one  
442 thread (serial computing) and two threads (parallel computing) and time  
443 limit (900s and 3600s) are not statistically significant.

#### 444 6.2. Heuristics evaluation

445 The twelve proposed methods ( $H_{11}$ ,  $H_{12}$ ,  $H_{21}$ ,  $H_{22}$ ,  $H_{31}$ ,  $H_{32}$ ,  $VND_{H_{11}}$ ,  
446  $VND_{H_{12}}$ ,  $VND_{H_{21}}$ ,  $VND_{H_{22}}$ ,  $VND_{H_{31}}$  and  $VND_{H_{32}}$ ) are now tested. As the  
447 proposed heuristics are not expected to find an optimal solution, the Relative  
448 percentage deviation (RPD), is measured for comparisons. We measure  $RPD$   
449 as follows: using the optimal solution or the best known solution, which is  
450 found through all heuristics and the MILP model ( $OPT_{best}$ ) and  $ALG_{SOL}$ ,  
451 which reports the makespan obtained by a given algorithm for a given in-  
452 stance:

$$453 \quad RPD = \frac{ALG_{SOL} - OPT_{best}}{OPT_{best}} \times 100$$

454  
455 Table 7 provides the summarized results of the MILP and the average  
456 algorithm deviations from the best known solution for the small instances.  
457 They are categorized by  $n$  and  $f$ .

458 As we can see in Table 7, it is clear that the mathematical model is unable  
459 to find an optimum or best solution for all the small instances considered. By  
460 increasing the number of jobs ( $n$ ) and by decreasing number of the factories  
461 ( $f$ ), the problem becomes harder for the MILP to solve. All VND algorithms  
462 perform better than the constructive algorithms.  $NR_2$  works better than the  
463 first one as a rule to assign jobs to factories. In order to know if the differences  
464 observed in Table 7 are statistically significant, a multifactor ANOVA of the  
465 results of the VND algorithms has to be done. The average RPD value for  
466 all the simple constructive heuristics is 6.75%, and this amount lowers to  
467

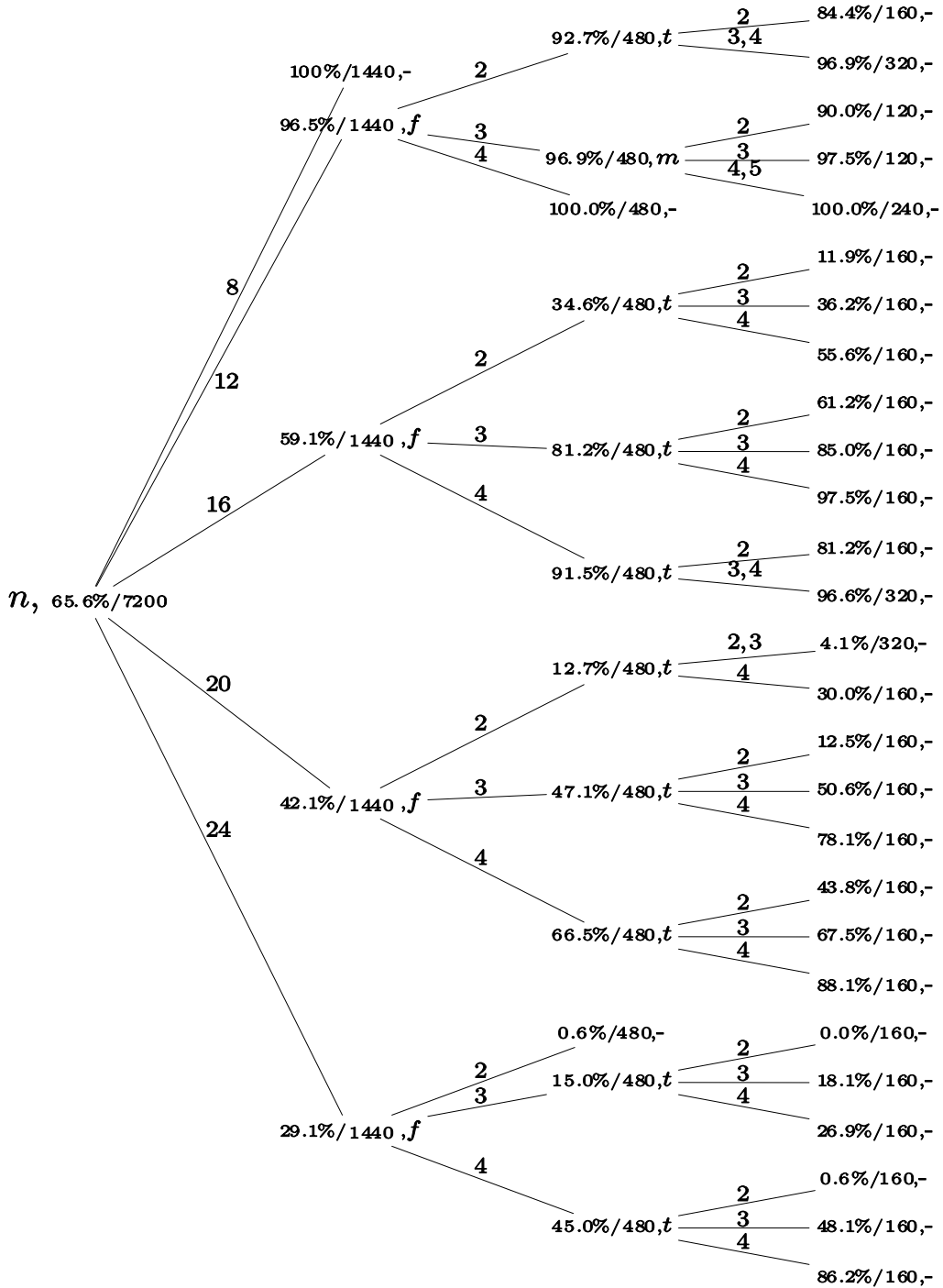


Figure 2: Decision tree for the MILP model evaluation.

| $f \times n$ | Algorithms |                 |                 |                 |                 |                 |                 |                               |                               |                               |                               |                               |                               |
|--------------|------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
|              | MILP       | H <sub>11</sub> | H <sub>12</sub> | H <sub>21</sub> | H <sub>22</sub> | H <sub>31</sub> | H <sub>32</sub> | VND <sub>H<sub>11</sub></sub> | VND <sub>H<sub>12</sub></sub> | VND <sub>H<sub>21</sub></sub> | VND <sub>H<sub>22</sub></sub> | VND <sub>H<sub>31</sub></sub> | VND <sub>H<sub>32</sub></sub> |
| 2 × 8        | 0.00       | 14.62           | 13.61           | 6.91            | 5.99            | 13.55           | 12.17           | 1.00                          | 0.76                          | 1.00                          | 0.76                          | 1.02                          | 0.78                          |
| 2 × 12       | 0.02       | 13.70           | 12.78           | 5.74            | 5.17            | 11.58           | 11.05           | 0.93                          | 0.87                          | 0.93                          | 0.87                          | 0.93                          | 0.87                          |
| 2 × 16       | 0.45       | 12.52           | 11.40           | 5.77            | 5.10            | 10.00           | 9.16            | 0.73                          | 0.55                          | 0.72                          | 0.53                          | 1.09                          | 0.53                          |
| 2 × 20       | 1.55       | 10.23           | 9.59            | 4.55            | 3.78            | 8.96            | 8.46            | 0.53                          | 0.36                          | 0.51                          | 0.37                          | 0.57                          | 0.37                          |
| 2 × 24       | 3.42       | 8.71            | 8.34            | 5.00            | 4.74            | 7.54            | 7.15            | 0.54                          | 0.21                          | 0.54                          | 0.21                          | 0.54                          | 0.21                          |
| 3 × 8        | 0.00       | 11.35           | 9.96            | 4.57            | 3.15            | 8.92            | 7.79            | 1.09                          | 0.70                          | 1.15                          | 0.76                          | 1.15                          | 0.76                          |
| 3 × 12       | 0.02       | 9.96            | 9.13            | 3.03            | 2.55            | 8.72            | 7.50            | 0.44                          | 0.28                          | 0.44                          | 0.28                          | 0.44                          | 0.28                          |
| 3 × 16       | 0.05       | 10.10           | 9.16            | 3.77            | 3.14            | 9.59            | 8.73            | 0.86                          | 0.56                          | 0.91                          | 0.56                          | 0.91                          | 0.56                          |
| 3 × 20       | 0.40       | 9.86            | 8.93            | 2.72            | 2.19            | 8.53            | 7.84            | 0.43                          | 0.43                          | 0.43                          | 0.43                          | 0.43                          | 0.43                          |
| 3 × 24       | 1.16       | 7.77            | 6.48            | 3.11            | 2.52            | 7.24            | 6.32            | 0.64                          | 0.33                          | 0.64                          | 0.33                          | 0.64                          | 0.33                          |
| 4 × 8        | 0.00       | 9.03            | 8.01            | 2.16            | 1.25            | 6.41            | 5.25            | 1.08                          | 0.63                          | 0.99                          | 0.63                          | 0.99                          | 0.63                          |
| 4 × 12       | 0.00       | 5.63            | 4.53            | 1.82            | 1.38            | 4.58            | 3.58            | 0.74                          | 0.47                          | 0.74                          | 0.47                          | 0.74                          | 0.56                          |
| 4 × 16       | 0.03       | 7.21            | 6.34            | 2.86            | 2.27            | 6.14            | 5.18            | 0.59                          | 0.28                          | 0.59                          | 0.28                          | 0.59                          | 0.28                          |
| 4 × 20       | 0.21       | 6.80            | 6.00            | 2.96            | 2.61            | 5.66            | 5.04            | 1.10                          | 0.63                          | 1.10                          | 0.63                          | 1.10                          | 0.63                          |
| 4 × 24       | 0.40       | 5.14            | 4.43            | 2.02            | 1.60            | 4.87            | 4.19            | 0.57                          | 0.26                          | 0.57                          | 0.26                          | 0.57                          | 0.26                          |
| Average      | 0.51       | 9.51            | 8.58            | 3.80            | 3.16            | 8.15            | 7.29            | 0.75                          | 0.49                          | 0.75                          | 0.49                          | 0.78                          | 0.50                          |

Table 7: Relative Percentage Deviation (RPD) of MILP and proposed algorithms over the best known solution for the small instances.

468 0.63% for the VND methods. The RPD factor difference between simple  
469 constructive heuristics and VND heuristics is very high. For this reason,  
470 we separated the statistical analysis in two ANOVAs: one for the simple  
471 heuristics and the other one for the VND methods. As explained before, there  
472 are 900 small instances, and each ANOVA considers six simple constructive  
473 heuristics or six VND methods with  $6 \times 900 = 5400$  data.

474 As with all parametric analyses, ANOVA requires some assumptions to  
475 be met. These are normality, homocedasticity and independence of residuals.  
476 While a slightly strong tailed normal distribution of the residuals is observed,  
477 residuals are clearly homoscedastic and independent, and according to the  
478 recent results of Basso et al. (2007) and Rasch and Guiard (2004), this is not  
479 a major problem. The response factor is again the RPD and the controlled  
480 factors are  $n$ ,  $m$ ,  $f$ ,  $t$  and *algorithms*. All the controlled factors in the ANOVA  
481 analysis, except  $m$  and  $t$  in six simple constructive heuristics, and except  $f$   
482 factor in six VND methods result in strong statistically significant differences  
483 in the RPD response variable, with  $p$ -values coming very close to zero. The  
484 results are not shown here due to reasons of space. In order to identify the  
485 best algorithm, the means plot and Tukey’s Honest Significant Difference

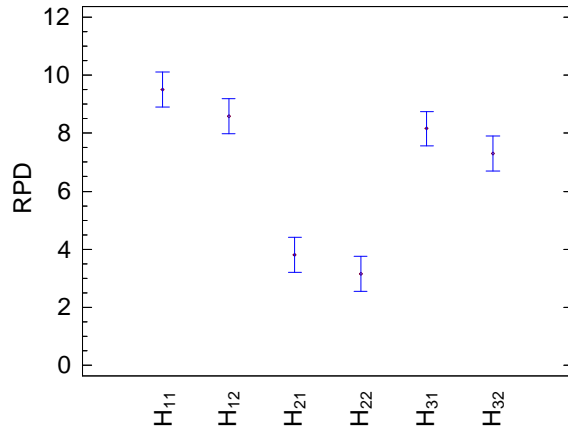


Figure 3: Means plot and 99% confidence level Tukey's HSD intervals for simple constructive heuristic methods and small instances.

486 (HSD) intervals (99% confidence) for the six simple constructive heuristics  
 487 and VND methods are shown in Figures 3 and 4, respectively.

488 As it is clear in Figure 3, the second heuristic performs better in compar-  
 489 ison with the other simple constructive heuristics and there is no significant  
 490 differences between the rules used to assign jobs to factories. However, it is  
 491 obvious in Figure 4 that the rules for allocating jobs to factories are impor-  
 492 tant, and  $NR_2$  is statistically different from  $NR_1$ . It is clear that the VND  
 493 algorithm almost improves all the initial solutions equally and that the kind  
 494 of initial solution to start the VND is not important for algorithms with the  
 495 same job assignment rule. No significant differences between the three VND  
 496 considered algorithms using  $NR_2$  is found.

497 The CPU times to solve small instances with the considered algorithms  
 498 are negligible; for example, the  $VND_{H_{32}}$  algorithm with 0.004693 seconds,  
 499 has the largest average consumed CPU time for the small instances.

### 500 6.3. Heuristics evaluation on large instances

501 In this case, for calculating the RPD, the best solution ( $OPT_{best}$ ) is the  
 502 best solution found among all twelve algorithms because, in large instances,  
 503 good MILP bounds are not known. A summarized result of the average RPD,  
 504 considering number of factories, number of products and number of jobs,  
 505 is shown in Table 8. Algorithms can be categorized into two groups: VND  
 506 algorithms,  $H_{21}$  and  $H_{22}$ , in one group, which perform better, and the rest

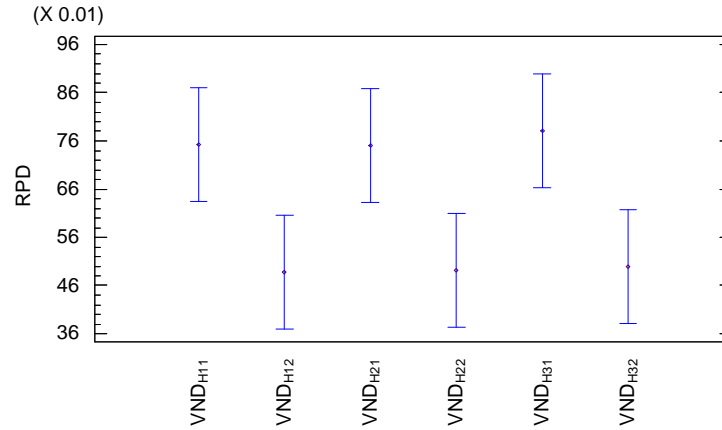


Figure 4: Means plot and 99% confidence level Tukey's HSD intervals for VND methods and small instances.

507 in another group. On the other hand, algorithms with  $NR_2$  work better than  
 508 those with  $NR_1$ .

509 The second group does not report good results if compared to the first  
 510 one, so it has been eliminated from the statistical analysis. A multifactorial  
 511 ANOVA has been carried out with only the first group to know if there are  
 512 any significant differences between results. Figure 5 shows a means plot (99%  
 513 confidence level Tukey's HSD intervals) for the first group of algorithms. It  
 514 is clear that the algorithms which use  $NR_2$  as a job assignment rule, report  
 515 better results. Moreover, the type of initial solution for the VND algorithms  
 516 does not play an important role. Finally, there is no significant difference  
 517 between the VND algorithms that use the same job allocation rule.

|                               |            | Algorithms      |                 |                 |                 |                 |                 |                               |                               |                               |                               |                               |                               |       |
|-------------------------------|------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------|
|                               |            | H <sub>11</sub> | H <sub>12</sub> | H <sub>21</sub> | H <sub>22</sub> | H <sub>31</sub> | H <sub>32</sub> | VND <sub>H<sub>11</sub></sub> | VND <sub>H<sub>12</sub></sub> | VND <sub>H<sub>21</sub></sub> | VND <sub>H<sub>22</sub></sub> | VND <sub>H<sub>31</sub></sub> | VND <sub>H<sub>32</sub></sub> |       |
| Relative Percentage Deviation | Factories  | 4               | 5.57            | 5.09            | 0.32            | 0.19            | 2.96            | 2.56                          | 0.06                          | 0.03                          | 0.05                          | 0.01                          | 0.05                          | 0.01  |
|                               | <i>(f)</i> | 6               | 3.77            | 3.29            | 0.11            | 0.06            | 1.64            | 1.31                          | 0.03                          | 0.01                          | 0.02                          | 0.00                          | 0.02                          | 0.00  |
|                               |            | 8               | 3.09            | 2.66            | 0.04            | 0.02            | 1.21            | 0.93                          | 0.02                          | 0.00                          | 0.01                          | 0.00                          | 0.01                          | 0.00  |
|                               |            | Aver            | 4.14            | 3.68            | 0.16            | 0.09            | 1.94            | 1.60                          | 0.04                          | 0.01                          | 0.03                          | 0.01                          | 0.03                          | 0.01  |
|                               | Products   | 30              | 3.78            | 3.34            | 0.21            | 0.11            | 2.23            | 1.86                          | 0.03                          | 0.01                          | 0.04                          | 0.01                          | 0.04                          | 0.01  |
|                               | <i>(t)</i> | 40              | 4.30            | 3.85            | 0.15            | 0.10            | 1.94            | 1.62                          | 0.04                          | 0.02                          | 0.02                          | 0.01                          | 0.02                          | 0.01  |
|                               |            | 50              | 4.36            | 3.85            | 0.11            | 0.05            | 1.65            | 1.32                          | 0.04                          | 0.01                          | 0.02                          | 0.00                          | 0.02                          | 0.00  |
|                               |            | Aver            | 4.14            | 3.68            | 0.16            | 0.09            | 1.94            | 1.60                          | 0.04                          | 0.01                          | 0.03                          | 0.01                          | 0.03                          | 0.01  |
|                               | Jobs       | 100             | 6.30            | 5.61            | 0.17            | 0.08            | 2.02            | 1.58                          | 0.05                          | 0.02                          | 0.03                          | 0.01                          | 0.03                          | 0.01  |
|                               | <i>(n)</i> | 200             | 3.76            | 3.28            | 0.15            | 0.07            | 1.92            | 1.55                          | 0.03                          | 0.01                          | 0.02                          | 0.00                          | 0.02                          | 0.00  |
| 500                           |            | 2.37            | 2.16            | 0.14            | 0.10            | 1.87            | 1.67            | 0.03                          | 0.01                          | 0.03                          | 0.01                          | 0.03                          | 0.01                          |       |
| Aver                          |            | 4.14            | 3.68            | 0.16            | 0.09            | 1.94            | 1.60            | 0.04                          | 0.01                          | 0.03                          | 0.01                          | 0.03                          | 0.01                          |       |
| CPU time (sec.)               | Factories  | 4               | 0.01            | 0.01            | 0.01            | 0.01            | 0.01            | 0.01                          | 4.39                          | 6.79                          | 2.90                          | 7.67                          | 2.55                          | 42.87 |
|                               | <i>(f)</i> | 6               | 0.01            | 0.01            | 0.01            | 0.01            | 0.01            | 0.01                          | 3.49                          | 7.73                          | 2.85                          | 8.94                          | 1.95                          | 6.11  |
|                               |            | 8               | 0.01            | 0.01            | 0.01            | 0.01            | 0.01            | 0.01                          | 3.26                          | 9.56                          | 1.86                          | 10.21                         | 1.83                          | 20.64 |
|                               |            | Aver            | 0.01            | 0.01            | 0.01            | 0.01            | 0.01            | 0.01                          | 3.71                          | 8.03                          | 2.54                          | 8.94                          | 2.11                          | 23.20 |
|                               | Products   | 30              | 0.01            | 0.01            | 0.02            | 0.02            | 0.01            | 0.01                          | 3.64                          | 8.05                          | 3.14                          | 11.00                         | 2.70                          | 45.20 |
|                               | <i>(t)</i> | 40              | 0.01            | 0.01            | 0.01            | 0.01            | 0.01            | 0.01                          | 3.59                          | 7.12                          | 2.45                          | 8.05                          | 1.96                          | 5.54  |
|                               |            | 50              | 0.01            | 0.01            | 0.01            | 0.01            | 0.01            | 0.01                          | 3.91                          | 8.91                          | 2.02                          | 7.77                          | 1.66                          | 18.88 |
|                               |            | Aver            | 0.01            | 0.01            | 0.01            | 0.01            | 0.01            | 0.01                          | 3.71                          | 8.03                          | 2.54                          | 8.94                          | 2.11                          | 23.20 |
|                               | Jobs       | 100             | 0.00            | 0.00            | 0.00            | 0.00            | 0.00            | 0.00                          | 1.09                          | 2.84                          | 0.27                          | 0.72                          | 0.24                          | 0.43  |
|                               | <i>(n)</i> | 200             | 0.00            | 0.00            | 0.00            | 0.00            | 0.00            | 0.00                          | 2.02                          | 3.85                          | 0.58                          | 2.22                          | 0.66                          | 1.37  |
| 500                           |            | 0.03            | 0.02            | 0.03            | 0.04            | 0.03            | 0.02            | 8.03                          | 17.39                         | 6.76                          | 23.88                         | 5.41                          | 67.81                         |       |
| Aver                          |            | 0.01            | 0.01            | 0.01            | 0.01            | 0.01            | 0.01            | 3.71                          | 8.03                          | 2.54                          | 8.94                          | 2.11                          | 23.20                         |       |

Table 8: Relative Percentage Deviation (RPD) and CPU times of proposed algorithms for the large instances.



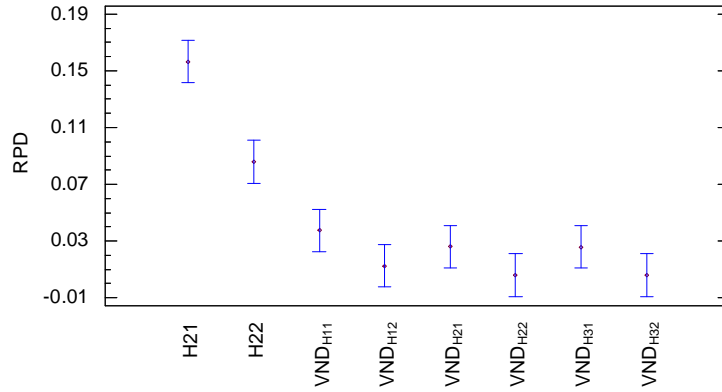


Figure 5: Means plot and 99% confidence level Tukey's HSD intervals for algorithms and large instances.

518 It is obvious that heuristic 2 performs better than heuristic 3 in both  
 519 small and large instances.

520 The interaction between algorithms and  $n$  has no significant effect on the  
 521 response variable. An increase in the number of machines always complicates  
 522 problems, thus there is no interest in showing these interactions. Interaction  
 523 between algorithms and the number of factories  $f$  is interesting. By increas-  
 524 ing the number of factories, the problem becomes easier, as it is shown in  
 525 Figure 6.

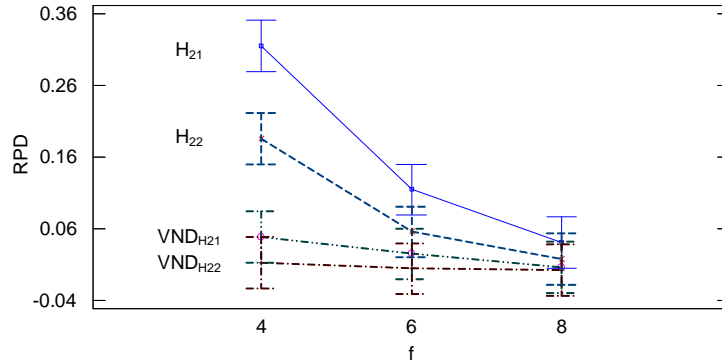


Figure 6: Means plot and 99% confidence level Tukey's HSD intervals for interaction between algorithms and number of factories  $f$  and large instances.

526 Neither the number of products nor the number of jobs factors have a  
 527 significant effect, and only an increase in either makes the problem easier to

528 solve for simple constructive algorithms. However, neither one has a signifi-  
529 cant effect on the VND algorithms.

530 In all the results, the RPD of VND<sub>H22</sub> is consistently lower than that  
531 of the other algorithms. Thus with more samples, it is expected that it will  
532 eventually become statistically better than the others. VND<sub>H22</sub> is better than  
533 VND<sub>H21</sub> because NR<sub>2</sub> checks all the factories when assigning a job and finally  
534 chooses the best one. It takes longer than NR<sub>1</sub>, which just places the job at  
535 the first available factory. However, when the number of factories increases,  
536 the algorithms that use NR<sub>1</sub> do not report good results.

537 The algorithms' CPU time consumption is summarized in Table 8. Simple  
538 constructive algorithms use a very short time in order to solve problems,  
539 while, as expected, the VND algorithms use more time if compared to simple  
540 constructive algorithms. VND<sub>H32</sub> consumes an average of 23.20 seconds, the  
541 longest CPU time consumption if compared to other algorithms. As Table 8  
542 shows, in the VND<sub>H32</sub> algorithm, factors  $n = 500$ ,  $t = 30$  and  $f = 4$  are the  
543 most CPU time consuming.

544 The VND methods try to improve the output of simple constructive al-  
545 gorithms and it is logical that take more time than simple constructive al-  
546 gorithms to solve problems. To compensate, VND algorithms report smaller  
547 RPD values than simple constructive algorithms. As Table 8 shows, the mini-  
548 mum RPD reported by a simple constructive algorithm is 9 times larger than  
549 the largest reported RPD by VND algorithms that use NR<sub>2</sub>.

550 If the quality of the solution is more important than CPU time con-  
551 sumption, then VND algorithms are the best options. Otherwise, a simple  
552 constructive algorithm can be a good choice when only CPU time consump-  
553 tion is more important. However, it is worth waiting a maximum time of  
554 almost 24 seconds to obtain a good solution. All the experimental results  
555 and the best solutions can be found at <http://soa.iti.es>.

## 556 7. Conclusion and future research

557 To the best of our knowledge, this paper is the first attempt to general-  
558 ize the Distributed Permutation Flowshop Scheduling Problem to the Dis-  
559 tributed Assembly Permutation Flowshop Scheduling Problem, where there  
560 is more than one production center to process jobs and a single assembly  
561 center to make final products from produced jobs. A mathematical model is  
562 presented and two solvers are used to solve it. Three constructive algorithms  
563 and three VND algorithms are proposed.

564 Computational evaluations were performed with two groups of small and  
565 large instances, and ANOVAs were used to analyze results. Results show  
566 that the VND algorithms report the best results. On the other hand, simple  
567 constructive algorithms consume little CPU time and still produce reasonable  
568 solutions.

569 For future works, setup times, transportation stages and distinct facto-  
570 ries can be considered for added realism. Other strategies can be used to  
571 construct VND neighborhoods. Other metaheuristics may report better so-  
572 lutions if compared to our proposed VND.

573

### 574 **Acknowledgements**

575 Rubén Ruiz is partially supported by the Spanish Ministry of Science and  
576 Innovation, under the project “RESULT - Realistic Extended Scheduling Us-  
577 ing Light Techniques” with reference DPI2012-36243-C02-01. Carlos Andrés  
578 is partially supported by the Spanish Ministry of Science and Innovation,  
579 under the project “INSAMBLE - Scheduling at assembly/disassembly syn-  
580 chronized supply chains” with reference DPI2011-27633.

581

### 582 **References**

- 583 Al-Anzi, F. S., Allahverdi, A., 2006. A hybrid tabu search heuristic for the two-  
584 stage assembly scheduling problem. *The International Journal of Operational*  
585 *Research*, 3 (2), 109–119.
- 586 Baker, K. R., 1974. *Introduction to sequencing and scheduling*. Wiley, New York.
- 587 Basso, D., Chiarandini, M., Salmaso, L., 2007. Synchronized permutation tests  
588 inreplicated  $i \times j$  designs. *Journal of Statistical Planning and Inference*, 137 (8),  
589 2564–2578.
- 590 Biggs, D., Ville, B. D., Suen, E., 1991. A method of choosing multiway partitions  
591 for classification and decision trees. *Journal of Applied Statistics*, 18 (1), 49–62.
- 592 Chan, F. T. S., Chung, S. H., Chan, P. L. Y., 2005. An adaptive genetic algorithm  
593 with dominated genes for distributed scheduling problems. *Expert Systems with*  
594 *Applications*, 29 (2), 364–371.
- 595 Chan, F. T. S., Chung, S. H., Chan, P. L. Y., 2006a. Application of genetic al-  
596 gorithms with dominant genes in a distributed scheduling problem in flexible  
597 manufacturing systems. *International Journal of Production Research* 44 (3),  
598 523–543.
- 599 Chan, F. T. S., Chung, S. H., Chan, P. L. Y., Finke, G., Tiwari, M. K., 2006b.  
600 Solving distributed FMS scheduling problems subject to maintenance: genetic  
601 algorithms approach. *Robotics and Computer-Integrated Manufacturing*, 22 (5-

- 602 6), 493–504.
- 603 Chen, S. H., Chang, P. C., Cheng, T. C. E., Zhang, Q., 2012. A self-guided ge-  
604 netic algorithm for permutation flowshop scheduling problems. *Computers &*  
605 *Operations Research*, 39 (7), 1450–1457.
- 606 Ching-Jong, L., Li-Man, L., 2008. Improved MILP models for two-machine flowshop  
607 with batch processing machines. *Mathematical and Computer Modelling*, 48 (7-  
608 8), 1254–1264.
- 609 Dong, X., Huang, H., Chen, P., 2009. An iterated local search algorithm for the  
610 permutation flowshop problem with total flowtime criterion. *Computers & Op-*  
611 *erations Research*, 36 (5), 1664–1669.
- 612 Framinan, J. M., Leisten, R., 2003. An efficient constructive heuristic for flowtime  
613 minimisation in permutation flow shops. *Omega, The International Journal of*  
614 *Management Science*, 31 (4), 311–317.
- 615 Framinan, J. M., Leisten, R., Ruiz-Usano, R., 2002. Efficient heuristics for flow-  
616 shop sequencing with the objectives of makespan and flowtime minimization.  
617 *European Journal of Operational Research*, 141 (3), 559–569.
- 618 Gao, J., Chen, R., 2011a. A hybrid genetic algorithm for the distributed permu-  
619 tation flowshop scheduling problem. *International Journal of Computational In-*  
620 *telligence Systems*, 4 (4), 497–508.
- 621 Gao, J., Chen, R., 2011b. An NEH-based heuristic algorithm for distributed per-  
622 mutation flowshop scheduling problems. *Scientific Research and Essays*, 6 (14),  
623 3094–3100.
- 624 Gao, J., Chen, R., Deng, W., Liu, Y., 2012a. Solving multi-factory flowshop prob-  
625 lems with a novel variable neighbourhood descent algorithm. *Journal of Com-*  
626 *putational Information Systems*, 8 (5), 2025–2032.
- 627 Gao, J., Chen, R., Liu, Y., 2012b. A knowledge-based genetic algorithm for per-  
628 mutation flowshop scheduling problems with multiple factories. *International*  
629 *Journal of Advancements in Computing Technology*, 4 (7), 121–129.
- 630 Hansen, P., Mladenovic, N., 2001. Variable neighborhood search: principles and  
631 applications. *European Journal of Operational Research*, 130 (3), 449–467.
- 632 Hariri, A. M. A., Potts, C. N., 1997. A branch and bound algorithm for the two-  
633 stage assembly scheduling problem. *European Journal of Operational Research*,  
634 103 (2), 547–556.
- 635 Jia, H. Z., Fuh, J. Y. H., Nee, A. Y. C., Zhang, Y. F., 2002. Web-based multi-  
636 functional scheduling system for a distributed manufacturing environment. *Con-*  
637 *current Engineering: Research and Applications*, 10 (1), 27–39.
- 638 Jia, H. Z., Fuh, J. Y. H., Nee, A. Y. C., Zhang, Y. F., 2007. Integration of genetic  
639 algorithm and Gantt chart for job shop scheduling in distributed manufacturing  
640 systems. *Computers & Industrial Engineering*, 53 (2), 313–320.
- 641 Jia, H. Z., Nee, A. Y. C., Fuh, J. Y. H., Zhang, Y. F., 2003. A modified genetic algo-

- 642 rithm for distributed scheduling problems. *Journal of Intelligent Manufacturing*,  
643 14 (3-4), 351–362.
- 644 Johnson, S. M., 1954. Optimal two- and three-stage production schedules with  
645 setup times included. *Naval Research Logistics Quarterly*, 1 (1), 61–68.
- 646 Kahn, K. B., Castellion, G. A., Griffin, A., 2004. *The PDMA handbook of new*  
647 *product development*, 2nd Edition. Wiley, New York.
- 648 Kass, G. V., 1980. An exploratory technique for investigating large quantities of  
649 categorical data. *Journal of Applied Statistics*, 29 (2), 119–127.
- 650 Koulamas, C., Kyparisis, G. J., 2001. The three stage assembly flowshop scheduling  
651 problem. *Computers & Operations Research*, 28 (7), 689–704.
- 652 Laha, D., Sarin, S. C., 2009. A heuristic to minimize total flow time in permutation  
653 flow shop. *Omega, The International Journal of Management Science*, 37 (3),  
654 734–739.
- 655 Lee, C. Y., Cheng, T. C. E., Lin, B. M. T., 1993. Minimizing the makespan in the  
656 3-machine assembly-type flowshop scheduling problem. *Management Science*,  
657 39 (5), 616–625.
- 658 Liu, H., Gao, L., 2010. A discrete electromagnetism-like mechanism algorithm for  
659 solving distributed permutation flowshop scheduling problem. *International Con-*  
660 *ference on Manufacturing Automation*, 156–163.
- 661 Moon, C., Kim, J., Hur, S., 2002. Integrated process planning and scheduling with  
662 minimizing total tardiness in multi-plants supply chain. *Computers & Industrial*  
663 *Engineering*, 43 (1-2), 331–349.
- 664 Morgan, J. N., Sonquist, J. A., 1963. Problems in the analysis of survey data, and  
665 a proposal. *Journal of the American Statistical Association*, 58 (302), 415–434.
- 666 Naderi, B., Ruiz, R., 2010. The distributed permutation flowshop scheduling prob-  
667 lem. *Computers & Operations Research*, 37 (4), 754–768.
- 668 Nawaz, M., Ensore, E. E., Ham, I., 1983. A heuristic algorithm for the m-machine,  
669 n-job flowshop sequencing problem. *Omega, International Journal of Manage-*  
670 *ment Science*, 11 (1), 91–95.
- 671 Pan, Q. K., Ruiz, R., 2012. Local search methods for the flowshop scheduling  
672 problem with flowtime minimization. *European Journal of Operational Research*,  
673 222 (1), 31–43.
- 674 Pan, Q. K., Ruiz, R., 2013. A comprehensive review and evaluation of permutation  
675 flowshop heuristics to minimize flowtime. *Computers & Operations Research*,  
676 40 (1), 117–128.
- 677 Pinedo, M., 2012. *Scheduling: Theory, Algorithms and Systems*, 4th Edition.  
678 Springer, New York.
- 679 Potts, C. N., Janov, S. V. S., Strusevich, V. A., Wassenhove, L. N. V., Zwaneveld,  
680 C. M., 1995. The two-stage assembly scheduling problem: Complexity and ap-  
681 proximation. *Operations Research*, 43 (2), 346–355.

- 682 Rasch, D., Guiard, V., 2004. The robustness of parametric statistical methods.  
683 *Psychology Science*, 46 (2), 175–208.
- 684 Ruiz, R., Andrés-Romano, C., 2011. Scheduling unrelated parallel machines with  
685 resource-assignable sequence dependent setup times. *International Journal of*  
686 *Advanced Manufacturing Technology*, 57 (5-8), 777–794.
- 687 Ruiz, R., Şerifoğlu, F. S., Urlings, T., 2008. Modeling realistic hybrid flexible flow-  
688 shop scheduling problems. *Computers & Operations Research*, 35 (4), 1151–1175.
- 689 Ruiz, R., Maroto, C., 2005. A comprehensive review and evaluation of permutation  
690 flowshop heuristics. *European Journal of Operational Research*, 165 (2), 479–494.
- 691 Ruiz, R., Stützle, T., 2007. A simple and effective iterated greedy algorithm for  
692 the permutation flowshop scheduling problem. *European Journal of Operational*  
693 *Research*, 177 (3), 2033–2049.
- 694 Stafford, E. F., Tseng, F. T., Gupta, J. N. D., 2005. Comparative evaluation of  
695 MILP flowshop models. *Journal of the Operational Research Society*, 56 (1),  
696 88–101.
- 697 Tozkan, A., Kirca, O., Chung, C. S., 2003. A branch and bound algorithm to min-  
698 imize the total weighted flowtime for the two-stage assembly scheduling problem.  
699 *Computers & Operations Research*, 30 (2), 309–320.
- 700 Tseng, F. T., Stafford, E. F., 2008. New MILP models for the permutation flowshop  
701 problem. *Journal of the Operational Research Society*, 59 (10), 1373–1386.
- 702 Vallada, E., Ruiz, R., 2010. Genetic algorithms with path relinking for the minimum  
703 tardiness permutation flowshop problem. *Omega, The International Journal of*  
704 *Management Science*, 38 (1-2), 57–67.
- 705 Vallada, E., Ruiz, R., Minella, G., 2008. Minimising total tardiness in the m-  
706 machine flowshop problem: A review and evaluation of heuristics and meta-  
707 heuristics. *Computers & Operations Research*, 35 (4), 1350–1373.
- 708 Vollmann, T. E., Berry, W. L., Whybark, D. C., Jacobs, F. R., 2005. *Manufacturing*  
709 *Planning and Control for Supply Chain Management*, 5th Edition. McGraw-Hill,  
710 Boston.
- 711 Wang, B., 1997. *Integrated product, process and enterprise design*, 1st Edition.  
712 Chapman & Hall, London.
- 713 Xu, X., Xu, Z., Gu, X., 2011. An asynchronous genetic local search algorithm for  
714 the permutation flowshop scheduling problem with total flowtime minimization.  
715 *Expert Systems with Applications*, 38 (7), 7970–7979.
- 716 Zhang, Y., Li, X., 2011. Estimation of distribution algorithm for permutation flow  
717 shops with total flowtime minimization. *Computers & Industrial Engineering*,  
718 60 (4), 706–718.
- 719 Zobolas, G. I., Tarantilis, C. D., Ioannou, G., 2009. Minimizing makespan in per-  
720 mutation flowshop scheduling problems using a hybrid metaheuristic algorithm.  
721 *Computers & Operations Research*, 36 (4), 1249–1267.