CIE 52 - October 29-31, 2025 - INSA Lyon, France.

Exact methods for the Distributed Permutation Flowshop Problem with transport considerations

Levi Ribeiro de Abreu[a], Bruno de Athayde Prata[a], Paz Perez-Gonzalez[b,*]

[a]*Department of Industrial Engineering, Federal University of Ceara, Ceara, Brazil*
[b]*Industrial Management, University of Seville, Seville, Spain*

Abstract

The consideration of transportation in the Distributed Permutation Flowshop Scheduling (DPFS) problem is one of the most interesting opportunities to include in this problem since more than one factory is taken into account to schedule the production. In this paper, the DPFS problem with transportation before processing is considered, modeled as a job and factory dependent release date. Three problems have been studied depending on the objective function: makespan, total tardiness and just-in-time. First, to frame the problem, different transportation considerations are analysed, and the related literature reviewed, trying to identify the different ways to include the transportation in the DPFS problem. Then, the proposed problem is tackled to be solved exactly, using different approaches: Mixed Linear Integer Programming (MILP) and Constraint Programming (CP) models. For each approach, two solvers have been tested: IBM CPLEX and Gurobi for the MILP model; and IBM CP Optimizer and Hexaly (using their respective modeling languages) for the CP model. An extensive computational experiment has been carried out to solve small and large-sized instances (within a time limit), in order to determine the efficiency of the solvers, the complexity of the objectives taking into account the difficulty of the methods to find optimal solutions, and the quality of the solutions in case that optimally is not guaranteed. The results shown that IBM CP is the most efficient solver, and the just-in-time is the most difficult objective to provide optimal solutions even for small-sized instances.

Keywords: Scheduling, Distributed Permutation Flowshop, transport before processing

1. Introduction

The (deterministic) Distributed Flowshop Scheduling (DFS) problem is receiving increasing attention since the seminal paper by Naderi and Ruiz [1] was published fifteen years ago. Since then, many contributions are appearing in this field of the Operational Research, increasing exponentially each year [2]. This layout consists of a number of factories, with a flowshop on each factory, where a set jobs should be processed. The decisions to be taken are the assignment of the jobs to the factories, and the sequence of the jobs for each factory. The most interesting feature of the DFS problem is the centralization of the scheduling decision of more than one factory in a supply chain context. The fact that the factories have different locations would imply movement of material (raw material, semi-processed product or final product), among them or even with other facilities as warehouses, distribution centers, retailers or the final customer (see Figure 1). Therefore, the scheduling decision without transportation information can be myopic, since the assignment of the jobs to the factory can be optimal in terms of the completion times of the jobs (when using the classical objectives as makespan, total completion time or even due dates related objectives as total tardiness), but not very good solution in terms of logistic costs, depending on factors such as from which supplier is coming the raw material to process the job in the assigned factory; or, to which warehouse or retailer is going the job after is processed in the assigned factory. Additionally, taking into account inter-factory transportation (i.e. semi-processed product transferred between factories) classical objectives as makespan can even been improved [3].

*Corresponding author
Email addresses: levi.abreu@ufc.br (Levi Ribeiro de Abreu), baprata@ufc.br (Bruno de Athayde Prata), pazperez@us.es (Paz Perez-Gonzalez)

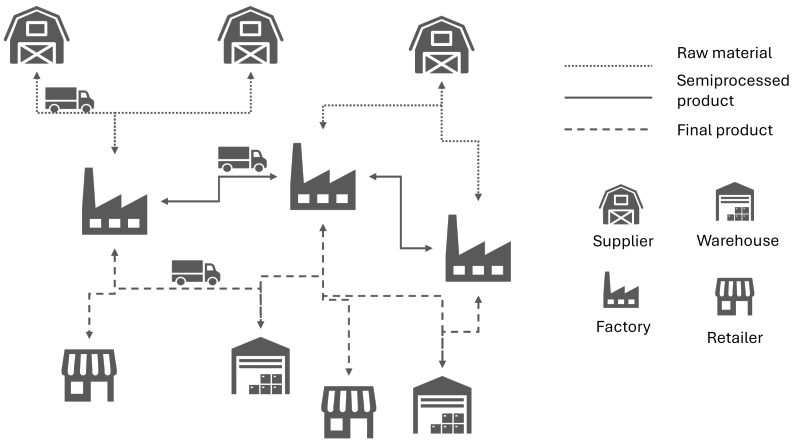

Figure 1: Transportation of material among facilities in a supply chain

In this paper, we consider the Distributed Permutation Flowshop Scheduling problem with transportation before processing. This means that the raw material to process each job should be transported from the suppliers to the assigned factory. This transportation time is modeled in the problem as a release date of the job, but depends on the factory too, extending the classic concept of release dates to the distributed scenario. The objectives taken into account are makespan, total tardiness and just-in-time. The three problems are NP-hard, due to the permutation flowshop scheduling problem (special case with only one factory) is NP-hard for the three objectives even when the release dates are equal to zero (except the makespan case when the number of machines are equal to two, but it is NP-hard with release dates) (see e.g [4]).

The NP-hardness does very complex to obtain optimal solutions in reasonable times, and the most of the DPFS literature, although provide the models of the problems, these models are not implemented and tested [2]. Additionally, the proposed problems have not been considered in the literature, so, as a starting point, the complexity is analyzed in this paper, with two different ways to solve exactly the problem: Constraint Programming (CP) and Mixed Integer Linear Programming (MILP) models, and implementing each method by two commercial solvers. On the one hand, the use of CP in production scheduling problems has several advantages, due to the use of logic programming in modeling, it is possible to represent scheduling problems with less constraints, using logical arguments, linear and non-linear operators and start and end constraints to each job operations (using to model delivery time to the factory or release dates presented in the proposed problem). Therefore, it is possible to solve the models with competitive computational costs [5]. For this methodology, two solvers have been used, modeling the problem with their own languages: IBM CP Optimizer and Hexaly. On the other hand, the MILP model is the most extended exact method to solve scheduling problems in the literature (see e.g. [1] where different models for the DPFS with makespan are tested, or [2] where the DPFS literature is reviewed showing the methods applied to the different problems). For this methodology, two solvers have been used: IBM CPLEX and Gurobi.

The structure of the paper is as follows: In Section 1, the proposed problem is detailed. The literature about DPFS with transportation consideration is reviewed in Section 3. The different models (the MILP model and the two CP modes for the two different modeling languages) are described in Section 4. The computational experimentation is carried out in Section 5. Finally, the conclusions are presented in Section 6.

2. Problem description

In the flowshop scheduling problem n jobs should be scheduled in m machines in series, i.e., each job implies m operations and the route of the jobs is the same, being processed in the first machine, then in the second machine and so on, taking into account the deterministic processing time for each job $j = 1, \ldots, n$ on each machine $i = 1 \ldots, m$, denoted as p_{ij}. The Distributed Flowshop Scheduling problem implies more than one factories ($f = 1, \ldots nf$, with

$nf \geq 2$) with a flowshop layout each, increasing the complexity of the problem due to the decisions: assigning the jobs to a factory and sequencing the assigned jobs in each factory.

In the general problem factories are identical, and each job can be assigned to only one factory, being all the factories eligible to process all jobs. Once the job is assigned to a factory, it should be processed on all machines of the factory (being the machines available at the beginning of the scheduling horizon), following the same route. Additionally, each job can be processed by only one machine at time, and each machine can process only one job at time. In particular, the Distributed Permutation Flowshop Scheduling (DPFS) problem includes the widely considered permutation constraint, imposing that, for all jobs assigned to each factory, the sequence to process them is the same for all the machines. For the general case, a solution (semiactive schedule) implies a sequence on each machine formed by the jobs assigned on each factory, and therefore, the problem has $\binom{n+nf-1}{n} \cdot (n!)^m$ possible solutions. The permutation constraint reduces the number of solutions of the flowshop scheduling problem from $(n!)^m$ to $n!$, and, consequently, to $\frac{(n+nf-1)!}{(nf-1)!}$ for the DPFS problem.

In general, the problem has been widely studied considering that the transportation times are irrelevant (outside as well as inside of the factories). This means that, once a job is assigned to a specific factory, the job is available to be processed at the beginning of the scheduling horizon, it cannot be transferred to another factory until its processing is completed, and the time needed to delivery it to the final customer is out of the scope of the problem. However, the advantages of the distributed approach is that it allows to include more supply chain information to the scheduling problems as transportation considerations. This integration of the decisions enriches and does more efficient the supply chain. In this paper, we consider the DPFS problem with transportation before processing, i.e., raw material to process each job (or semi-elaborated job) is not available in the factory at the beginning of the scheduling horizon, and it should be transported to the factory. This can be modeled as release dates that depend on the job and the factory (denoted r_{fj}), since the raw material (or the semi-elaborated job) should be transferred from the supplier to the factory where it should be processed. For each job j, its completion time, C_j, depends on the factory where it is assigned, and the position in the sequence of jobs assigned to the factory. In order to determine the influence of the transportation consideration, three objective functions are studied: makespan (C_{max}), defined as the maximum completion time of all jobs; the total tardiness ($\sum T_j$), where $T_j = \max\{C_j - d_j, 0\}$, being d_j the due date of the job j; and the named Just-In-Time (JIT) objective given by the expression $\sum |d_j - C_j|$. Using the $\alpha|\beta|\gamma$ Graham's notation [6], and the notation provided by [2] for the DFS, the three problems considered are $PFm|prmu, r_{fj}|C_{max}$, $PFm|prmu, r_{fj}|sumT_j$ and $PFm|prmu, r_{fj}|JIT$. In the following section, the literature about DPFS with transportation is reviewed, providing a framework for the problem studied in this paper.

3. Distributed Flowshop Scheduling problems with transportation consideration

The DPFS literature considering transportation is limited. Three different cases [2] are presented (see Figure 1):

- Transportation before processing (supplier-factory transportation). Jobs (or raw material to process the jobs) are located in a place different than the factory where they are going to be processed (assigned factory), and the transportation time from the warehouse or supplier should be taken into account in the scheduling decision.
- Transportation during processing (inter-factory transportation). Jobs can be transferred between factories while they are processed. In this case, each operation of the job can be carried out in different factories.
- Transportation after processing (factory-retailer/warehouse transportation). The final product should be transported to a warehouse or the retailer (customer) once it has been processed in the assigned factory.

To the best of our knowledge, the first work related about DPFS with transportation is [7], considering transportation before processing, eligibility constraint (i.e. all the jobs cannot be processed in all the factories) and a Pareto multi-objective approach to minimize makespan, maximum lateness and the total cost related to the transportation and factory setup, $PFm|prmu, M_j, tbp|\#(C_{max}, maxL_j, TC)$. The tbp is modeled as a release date dependent on the job and the factory, r_{fj}. The authors propose a MILP model to solve the problem exactly for small size instances. Then, they provide different methods to solve it approximately, including some constructive heuristics adapted from the literature, a NEH algorithm, and an adaptation of the NSGA-II as metaheuristic approach. Considering too transport before processing, [8] study the heterogeneous DPFS problem $RFm|prmu, tbp, no-wait|C_{max}$,

where the transportation is modeled as a release time dependent on the factory (r_f). The problem is described by a MILP, and solved by an ABC metaheuristic. The same idea about the transportation is considered by [9] for the problem $RFm|prmu, tbp, no-wait, s_{fijk}|C_{max}$, modeling the problem by a MILP, and developing an Artificial Bee Colony algorithm hybridized with Q-learning approach to solve it approximately.

Regarding transportation during processing, to the best of our knowledge, only [3] tackle this case, with an interesting analysis of the problem, analyzing its relationship with the classical Hybrid Flowshop Scheduling problem. They consider as objective the makespan, $PFm|tdp|C_{max}$, and it should be notice that the permutation constraint is lost, due to the movement of jobs among factories. The authors propose two approaches to solve exactly the problem, a MILP model and a CP model, and an Iterated Gready metaheuristic to solve it approximately, including some speed-ups for the makespan calculation.

Finally, two different approaches are identified for the DFSP problem with transportation after processing: The first approach is the transportation assumption considered by [10], consisting of allocating the jobs in batches to be delivered to their final customers (each batch is transported by a truck to its customer), and computing in the objective function the completion time as the delivery time of the batches. As objective function, they propose a multiobjective linear combination of makespan and TEC (total energy cost), including the velocity of the machines in the constraints which influence in the energy cost, $PFm|prmu, v_i, tap|F_l(C_{max}, TEC)$. The authors present a MIP model, and solve approximately the problem by a WOA metaheuristic. The sencond approach is more complex, integrating the scheduling problem as well the VRP, where the jobs are processed in a factory and then each truck should complete a route to deliver the jobs to each customer: the problem $PFm|prmu, tap|C_{max}$ is analyzed in [11], where the makespan is computed as the maximum delivery time. A MILP is provided, and the problem is approximately solved by a BWO metaheuristic. The problem $PFm|prmu, [d_j^-, d_j^+], tap|F_l(\sum w_j T_j, \sum w'_j E_j)$ is considered in [12], modeling the problem by a MILP, and a BSO metaheuristic is proposed to solve it approximately. [13] tackle a heterogeneous DPFS, $RFm|prmu, tap|\#(C_{max}, TEC)$, minimizing the maximum delivery times of the jobs to the customers, as well as the carbon emissions of the transportation. The problem is modeled by a MILP, and it is solved by an IGA. [14] consider again the problem $PFm|prmu, tap|C_{max}$ with a maximum capacity of the vehicles. The problem is approximately solved by a Hyper-heuristic algorithm with a GA embedded as intensification procedure. The no-wait constraint is considered in [15], but in this case the product should be delivered in a common warehouse (aggregation point) in the problem $PFm|prmu, tap, no-wait|C_{max}$, proposing a probability learning based multi-objective evolutionary algorithm. [16] tackle a similar problem but with a different objective, $PFm|prmu, tap, no-wait|TC$, where TC is the total cost including the processing and transportation costs. In their approach the authors compare two rules to load the trucks after processing, providing different transportation costs (that depend on fixed and variable costs). A metaheuristic approach based on hyper-heuristic and learning algorithms is tested. A fuzzy approach for the processing and transportation times is considered in [17], with the objective fuzzy carbon emissions and total costs (production and transportation). The transportation approach includes two phases: first the processed jobs are batched to be loaded in different vehicles and transported to a collection center where the jobs are assigned again to different vehicles to be transported to the final customers. An Ant Colony Optimization algorithm hybridized with hyper-heuristic is applied to solve the problem.

4. Problem modeling

In this section two different models are proposed: On the one hand, in Section 4.1, a Mixed Integer Linear Programming (MILP) model, based on the formulation for the problem $PFm|pmru|C_{max}$ by Naderi and Ruiz [1]. On the other hand, in Section 4.2, a Constraint Programming (CP) approach is presented, in this case modeling the problem by two modeling languages specifically designed for scheduling problems of different solvers: IBM CP Optimizer [18], and Hexaly [19]. For the IBM CP Optimizer modeling language, the model given by Naderi et al. [20] for the problem $PFm|prmu|C_{max}$ has been adapted. Additionally a model for the Hexaly CP modeling language is provided. Along the section, the same indexes and parameters are used in all the models. The decision variables of each case is included in its corresponding section.

Indices and sets

i: index for machines {1,2,...,m}

j: index for jobs {1,2,...,n}

k: index for positions in the sequence {1,2,...,n}
f: index for factories {1,2,...,nf}

Parameters

p_{ij}: processing time of job j on machine i
r_{fj}: release date of job j on factory f
d_j: due date of job j
$\mathcal{M}$: A big number for modeling purposes (the sum of all values of the p matrix)

4.1. Mixed-integer linear programming formulation

Decision variables and MILP model

C_{max}: maximum completion time of any job (makespan)
C_{kif}: Completion time of the job in position k on machine i at factory f
T_j: Tardiness of job j
E_j: Earliness of job j

$$X_{kjf} = \begin{cases} 1, & \text{if job } j \text{ will occupy position } k \text{ at factory } f \\ 0, & \text{otherwise} \end{cases}$$

$$\text{minimize} \quad C_{max} \text{ or } \sum_{k=1}^{n}\sum_{f=1}^{nf} T_{kf} \text{ or } \sum_{k=1}^{n}\sum_{f=1}^{nf}\left(E_{kf} + T_{kf}\right) \tag{1}$$

$$\text{subject to} \quad C_{kif} \geq C_{k,i-1,f} + \sum_{j=1}^{n} X_{jkf}.p_{ij}, \qquad \forall_{k,i>1,f} \tag{2}$$

$$C_{kif} \geq C_{k-1,i,f} + \sum_{j=1}^{n} X_{jkf}.p_{ij}, \qquad \forall_{k>1,i,f} \tag{3}$$

$$C_{k1f} \geq \sum_{j=1}^{n} X_{jkf} \cdot \left(p_{1j} + r_{fj}\right), \qquad \forall_{k,f} \tag{4}$$

$$C_{max} \geq C_{kmf}, \qquad \forall_{k,f} \tag{5}$$

$$\sum_{k=1}^{n}\sum_{f=1}^{nf} X_{jkf} = 1, \qquad \forall_{j} \tag{6}$$

$$\sum_{j=1}^{n}\sum_{f=1}^{nf} X_{jkf} = 1, \qquad \forall_{k} \tag{7}$$

$$T_{kf} \geq C_{kmf} - \sum_{j=1}^{n} X_{jkf} d_j - \mathcal{M}\left(1 - \sum_{j=1}^{n} X_{jkf}\right), \qquad \forall_{k,f} \tag{8}$$

$$E_{kf} \geq \sum_{j=1}^{n} X_{jkf} d_j - C_{kmf} - \mathcal{M}\left(1 - \sum_{j=1}^{n} X_{jkf}\right), \qquad \forall_{k,f} \tag{9}$$

$$C_{max} \geq 0, \qquad \forall_{j} \tag{10}$$

$$C_{kif} \geq 0, \qquad \forall_{k,i,f} \tag{11}$$

$$T_{kf} \geq 0, \qquad \forall_{k,f} \tag{12}$$

$$E_{kf} \geq 0, \qquad \forall_{k,f} \tag{13}$$

$$X_{kjf} \in \{0, 1\}, \qquad \forall_{k,j,f} \tag{14}$$

The objective functions (1) are the makespan or total tardiness or just-in-time minimization. Constraint set (2) calculates the tardiness for each job processed in position j. Constraint sets (2) and (3) determines the completion times of the jobs. Constraint (4) sets the completion time for the first job in the sequence. Constraint set (5) calculates the makespan. Constraint sets (6) and (7) are permutation constraints. The constraint sets (10), (11), and (14) establish the domain of the decision variables. If the objective function is the total tardiness minimization, the constraint sets

(8) and (12) are defined. Finally, if the objective function is the just-in-time minimization, the constraint sets (9) and (13) are defined.

4.2. Constraint programming models

Decision variables and CP Optimizer language model

$\mathcal{X}_{fij}$: is a unique optional interval variable for processing the operation of job j in machine i on factory f.

$\mathcal{Y}_{ij}$: is an interval variable to represents the operation duration of job j in machine i on selected factory.

Γ_{fi} is a sequence variable with an order of $\mathcal{X}_{fij}$ intervals variables processed in each machine i on each factory f.

$$\text{minimize} \quad \max_{\forall f,j} \texttt{endOf}\left(\mathcal{X}_{fmj}\right) \text{ or } \sum_{j=1}^{n} \max\left\{\texttt{endOf}\left(\mathcal{Y}_{mj}\right) - d_j, 0\right\} \text{ or } \sum_{j=1}^{n} \left|d_j - \texttt{endOf}\left(\mathcal{Y}_{mj}\right)\right| \tag{15}$$

$$\text{subject to} \quad \texttt{noOverlap}\left(\Gamma_{fi}\right), \qquad \forall_{fi} \tag{16}$$

$$\texttt{alternative}\left(\mathcal{Y}_{ij}, \left[\mathcal{X}_{fij}\right]_{f\in\{1,2,\ldots,nf\}}\right), \qquad \forall_{i,j} \tag{17}$$

$$\texttt{presenceOf}\left(\mathcal{X}_{f1j}\right) = \texttt{presenceOf}\left(\mathcal{X}_{fij}\right), \qquad \forall_{f,i>1,j} \tag{18}$$

$$\texttt{endBeforeStart}\left(\mathcal{X}_{fij}, \mathcal{X}_{fi+1j}\right), \qquad \forall_{f,i<m,j} \tag{19}$$

$$\texttt{sameSequence}\left(\Gamma_{fi}, \Gamma_{fi+1}\right), \qquad \forall_{f,i<m} \tag{20}$$

$$\text{interval } \mathcal{X}_{fij}, \quad \text{opt}, \quad \text{size} = p_{ij}, \quad \text{start} = \left(r_{fj}, INT\_MAX\right), \qquad \forall_{f,i,j} \tag{21}$$

$$\text{interval } \mathcal{Y}_{ij}, \qquad \forall_{ij} \tag{22}$$

$$\text{sequence } \Gamma_{fi}, \quad \text{on } \left[\mathcal{X}_{fij}\right]_{f\in\{1,2,\ldots nf\}:i\in\{1,2,\ldots,m\}:j\in\{1,2,\ldots,n\}}, \qquad \forall_{ij} \tag{23}$$

The objective function (15) are the makespan or total tardiness or just-in-time minimization. Constraints (16) imposes that a machine can process just one job at time. Constraints (17) state that a single operation of job in a machine can be processed in just on factory. Constraints (18) guarantees that once a job's operation on the first machine is allocated to a factory, all the other operations of the job must also be allocated to the same factory. Constraint (19) imposes that the next job operation on the next machine must start after the operation on the previous machine has ended (non-overlap job operation constraint). Constraints (20) imposes the permutation flow-shop characteristics with each machine has the same sequence of jobs operations. Finally, constraint sets (21), (22), and (23) define the scope of decision variables: The variables $\mathcal{X}_{fij}$ and $\mathcal{Y}_{ij}$ are interval decision variables with a duration and $\mathcal{X}_{fij}$ has a start time interval (scheduling horizon) between the release time r_{fj} and the maximum time available in the CP solver for the job to start processing INT_MAX, and Γ_{fi} is a sequence variable that stores the sequence of operations on each machine in each factory.

Decision variables and Hexaly language model

x_{ij}: is a interval variable for processing the operation of job j in machine i.

Υ_f is a sequence variable with the jobs sequence in each factory f.

$$\text{subject to} \quad x_{ij} = \texttt{interval}(0, \mathcal{M}), \qquad \forall_{i,j} \tag{24}$$

$$\Upsilon_f = \texttt{list}(n), \qquad \forall_f \tag{25}$$

$$\texttt{partition}\left(\Upsilon\right), \tag{26}$$

$$\texttt{lenght}\left(x_{ij}\right) = p_{ij}, \qquad \forall_{i,j} \tag{27}$$

$$\texttt{start}\left(x_{1j}\right) \geq r_{fj} \cdot \texttt{contains}\left(\Upsilon_f, j\right), \qquad \forall_{f,j} \tag{28}$$

$$x_{ij} > x_{i-1j}, \qquad \forall_{i,j} \tag{29}$$

$$\texttt{and}\left(k \in \{2, \ldots, \texttt{count}\left(\Upsilon_f\right)\}, x_{i\Upsilon_{fk}} < x_{i\Upsilon_{fk-1}}\right), \qquad \forall_{f,i} \tag{30}$$

$$\text{minimize} \quad \max_{\forall j} \texttt{end}\left(x_{mj}\right) \text{ or } \sum_{j=1}^{n} \max\left\{\texttt{end}\left(x_{mj}\right) - d_j, 0\right\} \text{ or } \sum_{j=1}^{n} \left|d_j - \texttt{end}\left(x_{mj}\right)\right| \tag{31}$$

Constraint sets (24) and (25) define the scope of decision variables. Constraint set (26) requires that each job be processed in only one factory. Constraint set (27) defines the scheduling duration of the operation of job j in machine i. Constraint set (28) imposes the transportation time before the start of job operations in a factory. Constraint set (29) imposes that the next job operation on the next machine must start after the operation on the previous machine has ended (non-overlap job operation constraint). Constraint set (30) imposes that the next job processed on a machine must start after the operation on the previous job in the same machine (non-overlap machine operation constraint). Finally, the objective function can be one among the makespan, total tardiness or just-in-time proposed in (31).

5. Computational results

As previously explained, and to the best of our knowledge, the problem under study has not been addressed previously in the literature. Therefore, the sets of instances of this section have been created using information of other set of instances for similar problems. The objective is to evaluate the complexity of the proposed models, and two set of instances are On the one hand, a small-sized set of instances has been generated, with $n \in \{4, 6, 8, 10, 12, 14, 16\}$, $m \in \{2, 3, 4, 5\}$, and $nf \in \{2, 3, 4\}$, and five instances for each combination of parameters, being a total of 420 instances. In this case the processing times have been generated by $p_{ij} \in U[1, 100]$. The release dates have been generated as suggested by [21], using a uniform distribution as well, $r_{fj} \in U[0, 5n]$. Finally, the due dates have been generated by $d_j = maxr_{fj} + \sum p_{ij} \cdot (1 + rand \cdot (1/nf) \cdot 0.5)$, with $rand \in U[0, 1)$. On the other hand, the most used testbed of the DPFS [1] has been extended to the proposed problem. The instances of this testbed can be considered as large-sized, and they are based on the well-known Taillard's testbed [22], with $n \times m \in \{20 \times 5, 20 \times 10, 20 \times 20, 50 \times 5, 50 \times 10, 50 \times 20, 100 \times 5, 100 \times 10, 100 \times 20, 200 \times 10, 200 \times 20, 500 \times 20\}$, and ten instances for each combination of parameters, resulting in 120 instances, combined with $nf \in \{2, 3, 4, 5, 6, 7\}$, providing a total of 720 instances. Taillard used a $U[1, 100]$ to generate the processing times in its instances, and the r_{fj} and d_j have been added using the same formulae that for the small-size instances.

Two different solvers have been used for the MILP model (Gurobi and IBM CPLEX), and, as previously explained, two solvers, using their own CP modeling languages have been used for the CP models (IBM CP Optimizer and Hexaly). The three objective functions have been used (Cmax, total tardiness and JIT). All the experiments have been computed in a set of computers with identical features. Computational times have been limited to 600 seconds, and a more than 1850 hours have been needed to obtain all the results.

For each objective function and each instance I solved by one of the methods $M \in$\{Gurobi,IBM CPLEX,IBM CP,Hexaly\}, it is reported the status (FEASIBLE, OPTIMAL, NOSOL_FOUND); the lower bound, LB_{IM}, being the optimal value of the objective function if this has been found (OPTIMAL status), or the best value found without optimally guaranteed (FEASIBLE status) and the time needed, CPU_I, (600 secs. in case of FEASIBLE or NOSOL_FOUND status). For each instance, the best/worst solution ($BestSol_I/Worst_I$) among the four methods is computed (it can be optimal or not), and the Relative Index Deviation (RDI) has been calculated with respect to the best solution with the following formulae: $RDI_{IM} = (LB_{IM} - BestSol_I)/(Worst_I - BestSol_I) * 100$.

Regarding the capacity of each method to solve the problems, Table 1 shows for each objective the percentage of instances for each status (on the left the small-sized instances, and on the right the large-sized instances). Regarding the small-sized instances, it can be observed that the 100% of them have been solved in less than 600 seconds by IBM CP only for the makespan. The complexity of the problem increases for the total tardiness, being the most complex case the JIT, since all the methods provide smaller percentages of optimally. Looking to the large-sized instances, the percentage of them optimally solved is negligible, but the CP methods are able to find feasible solutions for all the instances, being superior in this regard to the MILP model.

Additionally, the quality of the solutions can be observed by the ARDI (Average RDI) provided in Table 2 for small-sized instances and Table 3 for the large ones, showed for the different number of factories. For the small-sized instances, for makespan, IBM CP provides 0% of ARDI, being coherent with the status, but for the rest of the objectives, the ARDI is lower by IBM CPLEX. This means that, although IBM CP achieves higher percentages of optimal solutions, the quality of the feasible solutions are lower, increasing the ARDI. For large-sized instances, and consequently with the status results, IBM CP provides the best results, being the difference remarkable. It can be observed that GUROBI is not able to provide any solution when the number of factories is greater than four.

Table 1: Status for Small and Large-sized instances

	SMALL-SIZED INSTANCES				LARGE-SIZED INSTANCES			
	MILP		CP		MILP		CP	
STATUS	**GUROBI**	**IBM CPLEX**	**HEXALY**	**IBM CP**	**GUROBI**	**IBM CPLEX**	**HEXALY**	**IBM CP**
	Cmax				**Cmax**			
FEASIBLE	34.52%	13.10%	77.38%	0.00%	6.94%	89.86%	98.75%	87.36%
NOSOL_FOUND	0.00%	20.24%	0.00%	0.00%	93.06%	9.31%	0.00%	0.00%
OPTIMAL	65.48%	66.67%	22.62%	100.00%	0.00%	0.83%	1.25%	12.64%
	Total Tardiness				**Total Tardiness**			
FEASIBLE	54.05%	10.00%	86.67%	24.29%	6.94%	88.75%	98.61%	97.36%
NOSOL_FOUND	0.00%	40.00%	0.00%	0.00%	93.06%	10.42%	0.00%	0.00%
OPTIMAL	45.95%	50.00%	13.33%	75.71%	0.00%	0.83%	1.39%	2.64%
	JIT				**JIT**			
FEASIBLE	57.38%	11.67%	89.76%	31.19%	6.94%	88.89%	100.00%	98.89%
NOSOL_FOUND	0.00%	41.90%	0.00%	0.00%	93.06%	11.11%	0.00%	0.00%
OPTIMAL	42.62%	46.43%	10.24%	68.81%	0.00%	0.00%	0.00%	1.11%

Table 2: ARDI Small-sized instances

ARDI	**GUROBI**	**IBM CPLEX**	**HEXALY**	**IBM CP**
Cmax	**3.1883**	**1.7327**	**16.5260**	**0.0000**
2	0.6559	0.2927	19.2857	0.0000
3	3.7466	2.5456	19.2857	0.0000
4	5.1623	2.5688	11.0065	0.0000
Total Tardiness	**10.7242**	**1.0617**	**22.5216**	**1.3737**
2	5.0464	1.2673	27.1534	2.4020
3	13.0624	1.3580	20.0440	1.7193
4	14.0638	0.4887	20.3672	0.0000
JIT	**8.5716**	**0.7365**	**31.5224**	**5.1980**
2	5.1622	0.7246	36.8180	6.0252
3	8.6077	1.4286	30.7695	6.8367
4	11.9450	0.0000	26.9798	2.7320
Average	**7.4947**	**1.2367**	**23.5233**	**2.1906**

Table 3: ARDI Large-sized instances

ARDI	**GUROBI**	**IBM CPLEX**	**HEXALY**	**IBM CP**
Cmax	**35.1244**	**82.6817**	**57.1809**	**0.1391**
2	25.3952	65.2638	74.0065	0.6607
3	49.7182	79.2477	68.3104	0.0000
4		84.3405	60.8322	0.1736
5		88.8438	53.5468	0.0000
6		90.7364	44.7209	0.0000
7		88.2236	41.6688	0.0000
Total Tardiness	**18.9089**	**82.8398**	**49.1785**	**5.5760**
2	18.3743	70.3798	67.9438	2.7799
3	19.7109	76.9181	56.4334	7.4252
4		84.4704	52.5023	6.0541
5		86.4598	48.3769	5.8841
6		91.4512	38.2121	7.9167
7		89.0236	31.6026	3.3964
JIT	**23.7023**	**83.8169**	**46.9120**	**8.0873**
2	27.5786	73.2810	63.4257	3.7949
3	17.8878	81.9064	52.9842	10.3982
4		85.4879	46.7479	8.7472
5		83.0959	42.7424	8.7684
6		89.1042	38.7015	8.3333
7		91.1204	36.8706	8.4818
Average	**25.9119**	**83.1092**	**51.0905**	**4.6008**

6. Conclusions

In this paper, the Distributed Permutation Flowshop Scheduling (DPFS) problem with transportation before processing is considered, modeled as release dates depending on job and factory. Three objective functions have been considered: makespan, total tardiness and just-in-time minimization. Therefore, as a first step in the analysis of these new problems, they have been tackled by exact methods, analysing two methodologies: CP and MILP models, using two solvers for each one (IBM CPLEX and Gurobi for MILP, and IBM CP Optimizer and Hexaly for CP). The objective is to determine the most efficient exact approach to solve each problem, and compare the complexity of the problems regarding the objective function selected. An extensive experiment has been carried out, using small and large-sized instances. From the results, it can be derived that the most complex problem to be solved is the just-in-time minimization, followed by the total tardiness, and being the makespan the case that can be solve exactly for all the small-sized instances. Additionally, taking into account the solvers, the IBM CP Optimizer is the most efficient to find optimal solutions in less than 600 seconds regardless the objective function and the size of the instance. However, for small-sized instances, the MILP model solved by IBM CPLEX provides lowest RDI values for the total tardiness and the just-in-time, indicating that the feasible solutions provided by this solver have better quality in these very specific cases.

As future research lines, the results are going to be compared with the same problem but removing the release dates, in order to identify the influence of this parameter in the experiments. Additionally, the problems should be solved by approximate methods, to provide good solutions for the large-sized instances in reasonable computational times. Some of those presented in the literature may be adapted in order to identify their efficiency when the release date constraint is included.

Acknowledgments

This study was financed in part by the National Council for Scientific and Technological Development (CNPq), through grants 303594/2018-7, 309755/2021-2, and 407151/2021-4.

This research has been funded in part by Projects ENSURE - PID2022–142062OB-I00, funded by MCIN/AEI/10.13039 877 /501100011033/FEDER, EU, and ExPliCit - 101086465 – HORIZON-MSCA-2021-SE-01-01, funded by the European Commission

References

[1] B. Naderi, R. Ruiz, The distributed permutation flowshop scheduling problem, Computers and Operations Research 37 (2010) 754–768. doi:10.1016/j.cor.2009.06.019.

[2] P. Perez-Gonzalez, J. M. Framinan, A review and classification on distributed permutation flowshop scheduling problems, European Journal of Operational Research 312 (2024) 1–21. doi:10.1016/J.EJOR.2023.02.001.

[3] T. Becker, J. Neufeld, U. Buscher, The distributed flow shop scheduling problem with inter-factory transportation, European Journal of Operational Research 322 (2025) 39–55. doi:10.1016/j.ejor.2024.10.026.

[4] M. L. Pinedo, Scheduling: Theory, Algorithms and Systems, third ed., Springer Berlin / Heidelberg, 2008. doi:10.1073/pnas.0703993104.

[5] B. A. Prata, L. R. Abreu, M. S. Nagano, Applications of constraint programming in production scheduling problems: A descriptive bibliometric analysis, Results in Control and Optimization 14 (2024) 100350.

[6] R. L. Graham, E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, Optimization and heuristic in deterministic sequencing and scheduling: a survey, Annals of Discrete Mathematics 5 (1979) 287–326.

[7] S. Cai, K. Yang, K. Liu, Multi-objective optimization of the distributed permutation flow shop scheduling problem with transportation and eligibility constraints, Journal of the Operations Research Society of China 6 (2018) 391–416. doi:10.1007/s40305-017-0165-3.

[8] H. Li, X. Li, L. Gao, A discrete artificial bee colony algorithm for the distributed heterogeneous no-wait flowshop scheduling problem, Applied Soft Computing 100 (2021) 106946. doi:10.1016/j.asoc.2020.106946.

[9] F. Zhao, Z. Wang, L. Wang, A reinforcement learning driven artificial bee colony algorithm for distributed heterogeneous no-wait flowshop scheduling problem with sequence-dependent setup times, IEEE Transactions on Automation Science and Engineering 20 (2023) 2305–2320. doi:10.1109/TASE.2022.3212786.

[10] Q. Li, J. Li, X. Zhang, B. Zhang, A wale optimization algorithm for distributed flow shop with batch delivery, Soft Computing 25 (2021) 13181–13194. doi:10.1007/s00500-021-06099-0.

[11] Y. Fu, Y. Hou, Z. Chen, X. Pu, K. Gao, A. Sadollah, Modelling and scheduling integration of distributed production and distribution problems via black widow optimization, Swarm and Evolutionary Computation 68 (2022) 101015. doi:10.1016/j.swevo.2021.101015.

[12] Y. Hou, Y. Fu, K. Gao, H. Zhang, A. Sadollah, Modelling and optimization of integrated distributed flow shop scheduling and distribution problems with time windows, Expert Systems with Applications 187 (2022) 115827. doi:10.1016/j.eswa.2021.115827.

[13] S. Schulz, M. Schönheit, J. S. Neufeld, Multi-objective carbon-efficient scheduling in distributed permutation flow shops under consideration of transportation efforts, Journal of Cleaner Production 365 (2022) 132551. doi:10.1016/J.JCLEPRO.2022.132551.

[14] W. Chen, B. Qian, R. Hu, S. Zhang, Y. Wang, Hybrid hyper-heuristic algorithm for integrated production and transportation scheduling problem in distributed permutation flow shop, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 14086 LNCS (2023) 85–96. doi:10.1007/978-981-99-4755-3_8/TABLES/1.

[15] Z. Ding, Z. Li, B. Qian, R. Hu, C. Zhang, Probability learning based multi-objective evolutionary algorithm for distributed no-wait flow-shop and vehicle transportation integrated optimization problem, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 14086 LNCS (2023) 156–167. doi:10.1007/978-981-99-4755-3_14/TABLES/1.

[16] Z. Li, L. Bai, B. Qian, Y. Chen, Active learning based hyper-heuristic for the integration of production and transportation: A third-party logistics perspective, Computers and Industrial Engineering 194 (2024) 110381. doi:10.1016/J.CIE.2024.110381.

[17] J. H. Ma, R. Hu, N. Guo, B. Qian, Collaborative hyper-heuristic ant colony algorithm for solving multi-objective fuzzy low-carbon distributed permutation flow-shop and two-echelon vehicle transportation integrated scheduling problem, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 14862 LNCS (2024) 324–335. doi:10.1007/978-981-97-5578-3_26/FIGURES/4.

[18] P. Laborie, J. Rogerie, P. Shaw, P. Vilím, Ibm ilog cp optimizer for scheduling: 20+ years of scheduling with constraints at ibm/ilog, Constraints 23 (2018) 210–250. doi:10.1007/S10601-018-9281-X/FIGURES/20.

[19] L. Blaise, Modeling scheduling problems with hexaly, in: 11th IFAC Conference on Manufacturing Modelling, Management and Control – IFAC MIM2025, 2025.

[20] B. Naderi, R. Ruiz, V. Roshanaei, Mixed-integer programming vs. constraint programming for shop scheduling problems: new results and outlook, INFORMS Journal on Computing 35 (2023) 817–843.

[21] T. Ren, X. Wang, T. Liu, C.-C. Wu, D. Bai, L. Lin, M. Guo, Exact and metaheuristic algorithms for flow-shop scheduling problems with release dates, Engineering Optimization 54 (2022) 1853–1869.

[22] E. Taillard, Benchmarks for basic scheduling problems, European Journal of Operational Research 64 (1993) 278–285.