# A framework for production and outbound distribution: manufacturer dominates

Hugo Chevroton
*Université de Tours*
*LIFAT, EA 6300, ERL CNRS ROOT 6305*
Tours, France
hugo.chevroton@etu.univ-tours.fr

Jean-Charles Billaut
*Université de Tours*
*LIFAT, EA 6300, ERL CNRS ROOT 6305*
Tours, France
jean-charles.billaut@univ-tours.fr

Sonja U.K. Rohmer
Operations Research and Logistics
*Wageningen University*
Wageningen, Netherlands
sukrohmer@outlook.com

June 8, 2018

**Abstract**

In this paper we consider a two-level supply chain problem composed of a manufacturer and a third part logistic provider (3PL provider) at the operational level. The manufacturer operates a flow-shop for which inventory costs are considered. Customer deliveries are carried out by the 3PL provider for which routing are considered. Both agents face tardiness penalty costs, paid for late delivery by the manufacturer to the customer and by the 3PL provider to the manufacturer.

A scenario is proposed where the manufacturer dominates the negotiation with the 3PL provider, imposing the composition of batches for vehicles, the vehicle departure dates and estimating a delivery time. If the 3PL provider respects the estimated delivery times, there is no penalty, otherwise, some tardiness penalty costs apply. A Mixed Integer Linear Programming Model and three metaheuristic algorithms are proposed for this specific scenario. Some random data sets are generated and the quality and efficiency of the different methods are compared on a number of randomly generated instances in order. These experimentations highlight the superiority of specific methods on certain instances.

**Keys Words :** supply chain, multi-agent, manufacturer, 3PL provider, scheduling, vehicle routing,

cooperation

# 1 Introduction and literature review

In todays business environment, defined by fierce competition and increasing customer expectations, companies must find solutions for providing competitive services in terms of cost and quality. Production and distribution represent the main activities of a lot of companies and good scheduling of these activities may represent a great monetary savings potential. In this environment, these services are usually performed by different agents, which work together but serve their own interests. The traditional approach is to consider the production and the distribution problems separately, in order to optimize the gain of each agent, without taking into consideration possible negotiations between them. The goal of this paper is to treat these two problems of production and distribution planning in an integrated framework and investigate one collaboration scenario between the two agents.

While, since decades, there is a large amount of literature on each of the individual problems of production and distribution, the interest of the community for problems integrating both aspects is more recent. This interest is motivated by the possibility of making substantial savings in industries by addressing the two-level problem within a common framework than both problems separately.

A classification of such an integrated problem is proposed in [Chen, 2010]. The author provides a classification for integrated models based on different criteria such as for example the structure of the integration. They differentiate between three different kinds of integration: production and outbound transportation, inbound transportation and production and inbound transportation, production and outbound transportation.

In the survey [Moons et al., 2017], the authors present, define and classify different kinds of *production scheduling-vehicule routing problems (PS-VRP)*. This class of problems is presented first as IPODS (Integrated production and outbound distribution scheduling) problems by [Chen, 2010]. However, [Moons et al., 2017] focus only on problems where a VRP is solved for the delivery and thus exclude, for example, problems with single customer, direct shipment or use other transportation modes (such as airplane). The main classification criteria of this paper is related to the scheduling environment. Problems are thus separated according to their machine environment (single machine, parallel machines, job/flow shop) and presented according to other characteristics related to production, inventory and distribution.

In the following, we refer to papers in the literature presenting important similarities with our problem or resolution techniques.

In [Geismar et al., 2008], the authors integrate production and distribution problems in order to respect the lifespan of products from the end of production up to the delivery at the clients using one capacitated vehicle is used for delivery. Integrated in a genetic algorithm, the authors apply the split algorithm developed

by Prins [Prins, 2004] to optimally solve the VRP associated to a fixed sequence of jobs in the production.

[Ullrich, 2013] integrates production and outbound distribution scheduling in order to minimize the total tardiness. The first problem is a parallel machines scheduling problem. The second problem is a delivery problem of jobs with a fleet of capacitated vehicles. Delivery time windows, service times and destinations are considered. A Genetic Algorithm is proposed to solve the integrated problem. Inventory holding costs are not considered in this paper.

In [Condotta et al., 2013], the authors use a Tabu algorithm to deliver multiple products to the client respecting release and due dates. The objective is to find the optimal sequencing and the optimal batching of jobs.

In [Viergutz and Knust, 2014] the authors consider a single production facility modeled by a single machine. After completion, jobs have to be delivered to the customer by a single vehicle before a given deadline (perishability) and within a specified time window. Because of the limited resources, not all the customers may be supplied and the problem is to find a selection of customers in order to maximize the total satisfied demand. The authors propose a branch-and-bound algorithm. Inventory costs are not considered.

In [Cakici et al., 2014], the authors model the manufacturer production site as a parallel machine environment. A single capacitated vehicle is used for the delivery of jobs. The authors introduce weights associated to the jobs in order to take inventory holding costs into account. The objective function is to minimize the total weighted delivery time. The authors propose a MILP model and heuristic resolution approaches.

In [Fu, 2014] the author proposes different scheduling problems with different routing hypotheses. The problems are based on the coordination and domination scenarios between agents and the author develops different models and resolution methods according to the collaboration scenario. No inventory cost is considered.

In [Rohmer et al., 2015], the authors describe a production problem with outbound distribution, including inventory costs. This paper proposes models for a scenario with dominance of the 3PL provider (which imposes the number of vehicles and departure dates to the manufacturer) and proposes research directions by suggesting different coordination scenarios for the two agents.

A problem including inbound and outbound transportation from both sides of the production stage is treated in [Koç and Sabuncuoglu, 2017]. Inventory costs are considered and the objective is to schedule the jobs on the machines and the vehicles so that the sum of transportation and inventory holding costs are minimized. The authors propose a MILP model, a lower bound and heuristic algorithms for solving large size instances.

In [Kergosien et al., 2017] the authors propose a problem inspired by the chemotherapy production and delivery environment. Drugs are produced on parallel machine and delivered on multiple distant sites by an unique agent. The model include soft due date for the delivery to customers and hard lifespan for

products. The Bender decomposition is adapted. The resolution method and uses the delivery problem as master problem and a relaxed version of the production problem as slave. Cuts, upper and lower bounds are implemented to improve the model. This method shows considerable improvements compared to a classic model resolution with CPLEX given a 30 minutes runtime and instances with up to 40 jobs.

The research presented in this paper builds on this work. We propose a new scenario with a strong dominance of the manufacturer, who imposes the composition of batches for each vehicle, set departure dates, and evaluates delivery times. To evaluate the delivery dates, the MILP model of the manufacturer incorporates a routing phase. This makes the problem more difficult to solve, even if the sequencing of jobs in each batch follows a given rule (EDD order).

Section 2 gives a formal definition of the problem and introduces the notations. A Mixed Integer Linear Programming (MILP) model is presented for the problem. In Section 3, different resolution methods (GRASP, Tabu Search and Genetic Algorithm) are described and decomposed on different level. In section 4, we present the results of the resolution methods on randomly generated instances. Section 5 concludes the paper and gives some directions for future research.

## 2 A multi agent model

This research aims to propose an integrated model for a supply chain problem, where a manufacturer and a 3PL-provider are involved. For the manufacturer, we consider several costs (inventory costs, transportation costs, penalty costs, ...) that are related to the production phase and the transportation phase, in order to obtain a more realistic model. Despite inventory costs play a major role in production planning, they are often neglected in production scheduling models. Two kinds of inventory costs are considered: work in process (WIP) inventory and finished products inventory (FIN).

More precisely, this multi-agent problem consists of two sub-problems: a permutation flow-shop scheduling problem for the production problem of the manufacturer and a vehicle routing problem for the 3PL provider. According to the five-field notation $\alpha|\beta|\pi|\delta|\gamma$ proposed in [Chen, 2010], the notation of the problem that we consider is

$$Fm||V(\infty,\infty), routing|n|\gamma$$

where $Fm$ indicates an $m$-machine flow shop; $V(\infty,\infty)$ means that a sufficient number of vehicles are available and the capacity of each vehicle is unbounded; *routing* means that orders going to different customers can be transported in the same shipment, and a vehicle routing problem has to be solved; $n$ indicates that each job belongs to one customer. Finally, $\gamma$ is the objective function, detailed later in this section.

## 2.1 Production problem

The production problem considered in this paper is an $m$-machine permutation flow shop with work in process (WIP) and finished product inventory considerations. We suppose that we have a set $\{J_1, ..., J_n\}$ of $n$ jobs to schedule. A manufacturer is looking to design a production schedule for all jobs on the machines. Each job $J_j$ has a given processing time $p_{i,j}$ on machine $M_i$. After the completion of a job on a machine, a work-in-process inventory is generated, before the processing of the job on the next machine. The holding cost for this inventory depends on the job and is denoted by $h_j^{WIP}$ for job $J_j$. For sake of clarity, we do not assume that the work-in-process holding cost depends on the machines. After the completion of the job on the last machine, the finished product inventory is kept until the departure of the job for delivery. The holding cost for the finished product inventory depends on the product and is denoted by $h_j^{FIN}$ for job $J_j$. Job $J_j$ has to be delivered to customer $j$ for a given due date denoted by $d_j$. A delay in the delivery of the product generally results in a loss for the manufacturer, that can be financial, hence the manufacturer incurs a penalty cost $\pi_j^M$ per time unit of late delivery of $J_j$, that is paid to the customer.

We denote by $C_{i,j}$ the completion time of job $J_j$ on machine $M_i$, $1 \leq i \leq m$. In our model, the manufacturer decides the composition of vehicles (i.e. the batches) and their departure dates. Furthermore, the manufacturer estimates a delivery date for each job, assuming that the jobs are delivered in EDD order. Therefore, he has an estimation of the amount of the penalty costs due to the customers, which is called a Pseudo Penalty Cost. The objective of the manufacturer is to minimize his total cost, which is composed by the inventory cost $IC$, a vehicle cost $VC$ (related to the number of vehicles used) and the pseudo penalty costs $PPC^M$. At this step, the manufacturer does not know the real delivery dates.

We denote by $IC$ the total inventory cost. Its expression is the following:

$$IC = \sum_{j=1}^{n} (C_{m,j} - C_{1,j}) q_j h_j^{WIP} + \sum_{j=1}^{n} (f_j - C_{m,j}) q_j h_j^{FIN} \tag{1}$$

where $f_j$ denotes the departure time of $J_j$ and $q_j$ is the quantity of items of job $J_j$.

We denote by $PC^M$ the penalty cost for tardiness. The expression of $PC^M$ is:

$$PC^M = \sum_{j=1}^{n} \pi_j^M T_j^M \tag{2}$$

Remember that $T_j^M$ depends on the delivery dates that will be given by the 3PL provider (denoted $D_j^{3PL}$ for $J_j$) and that are not known at the moment. For this reason, the manufacturer considers a Pseudo Penalty Cost $PPC^M$ involving an estimation of the tardiness $PT_j^M$ of $J_j$, based on an estimation of the delivery time denoted $D_j^M$ of $J_j$, defined by: $PT_j^M = \max(0, D_j^M - d_j)$, where $D_j^M$ is determined assuming that the jobs in a batch are delivered in EDD order. We have:

$$PPC^M = \sum_{j=1}^{n} \pi_j^M PT_j^M \tag{3}$$

We denote by $VC$ the vehicle cost. This cost is defined by:

$$VC = c^V V \tag{4}$$

where $c^V$ is the cost of one vehicle and $V$ is the number of vehicles required by the manufacturer.

The pseudo total cost for the manufacturer is given by:

$$PTC^M = IC + PPC^M + VC \tag{5}$$

## 2.2 Distribution problem

The distribution is done by a third-party-logistics (3PL) provider. For each batch of jobs to deliver, he wants to find an optimal route for the delivery of products from the manufacturer site to the multiple customer locations. Two hypotheses are considered:

1. Vehicles stay at the manufacturing site and their departure times are imposed by the manufacturer.
2. A vehicle takes all the jobs that are available (completed).

Assuming this condition, we can have at most $n$ potential departure times, with the departure time of vehicle $k$ denoted by $F_k$, $k \in \{1, ..., n\}$, given by the manufacturer.

We denote by $t_{i,j}$ the travel time between site $i$ and site $j$ ($i, j \in \{0, ..., n+1\}$), where site 0 corresponds to the manufacturer site, site $n+1$ corresponds to the depot of the 3PL provider and site $j$ corresponds to the site of the customer waiting for job $J_j$ ($j \in \{1, ..., n\}$).

For each trip, the 3PL provider bears the costs for the routing, denoted by $RC_k$ for the route of vehicle $k$, which depends on the total travel time and a penalty cost per time unit to the manufacturer if the final delivery date ($D_j^{3PL}$) is greater than $D_j^M$. The total routing cost is equal to:

$$RC = \sum_{k=1}^{V} RC_k \tag{6}$$

The routing cost is the sum of the costs for all the routes, leaving from the manufacturer location and returning to the depot of the 3PL provider. The penalty cost is denoted by $PC^{3PL}$. We assume that this function is related to the total tardiness, i.e. we denote by $T_j^{3PL}$ the tardiness of delivery of $J_j$ from the point of view of the 3PL provider (related to the due date $D_j^M$ estimated by the manufacturer): $T_j^{3PL} = \max(0, D_j^{3PL} - D_j^M)$. $PC^{3PL}$ is defined by:

$$PC^{3PL} = \sum_{j=1}^{n} \pi_j^{3PL} T_j^{3PL} \tag{7}$$

The total cost for the 3PL provider is given by:

$$TC^{3PL} = RC + PC^{3PL} - VC \tag{8}$$

## 2.3 Integrated Problem

The two problems outlined above are interconnected and dependent on each other. We assume that:

- the number of vehicles $V$,
- the dates of departure of the vehicles ($F_k$, $1 \leq k \leq V$),
- the first estimation of the completion delivery date $D_j^M$,

are determined by the manufacturer during the construction of his schedule, minimizing $PTC^M$. The vehicles of the 3PL provider take the jobs and distribute them to the customers at minimum cost. The 3PL provider minimizes $TC^{3PL}$ costs. Then, the 3PL provider gives the delivery dates to the manufacturer with the implications on tardiness, so that the real penalty cost $PC^M$ for the manufacturer can be computed.

The real cost for the manufacturer and for the 3PL provider are:

$$TC^M = IC + PC^M + VC - PC^{3PL} \tag{9}$$

$$TC^{3PL} = RC + PC^{3PL} - VC \tag{10}$$

The whole process is described more formally in Alg. 1.

**Algorithm 1** General framework

The manufacturer optimizes his production schedule: MIN $PTC^M$

*// From the schedule we deduce the jobs completion times, the number of vehicles used, the batch compositions, their departure times and an estimation $D_j^M$ of the delivery completion times*

**for** $k$ **in** 1 **to** $V$ **do**

The 3PL provider delivers the jobs optimally to minimize his costs $RC_k + PC_k^{3PL}$.

*// From the routing of vehicle k we deduce the delivery dates of jobs $D_j^{3PL}$*

**end for**

Compute the total cost of the 3PL provider: $TC^{3PL} = \sum_{k=1}^{V} RC_k + PC_k^{3PL} - VC$

Compute the total cost of the manufacturer: $TC^M = IC + PC^M - PC^{3PL}$+VC

## 2.4 Mixed Integer Linear Programming models

We propose in this section a MILP model for solving the manufacturer problem and a MILP model for the 3PL provider.

The data used in the problem are defined in the following.

| | |
|---|---|
| $n$ | number of jobs |
| $m$ | number of machines |
| $p_{i,j}$ | processing time of job $J_j$ on machine $M_i$, $1 \leq j \leq n$, $1 \leq i \leq m$ |
| $q_j$ | quantity of items of job $J_j$, $1 \leq j \leq n$ |
| $d_j$ | delivery due date of job $J_j$, $1 \leq j \leq n$ |
| $t_{s_1,s_2}$ | travel time between site $s_1$ and site $s_2$, $s_1, s_2 \in \{0, 1, ..., n+1\}$ |
| $M$ | an arbitrary high value |

The costs that have to be taken into account are the following.

| | |
|---|---|
| $h_j^{WIP}$ | holding cost for WIP inventory of job $J_j$, $1 \leq j \leq n$ |
| $h_j^{FIN}$ | holding cost for finished product inventory of job $J_j$, $1 \leq j \leq n$ |
| $\pi_j^M$ | penalty cost of the manufacturer for late delivery of job $J_j$, $1 \leq j \leq n$ |
| $c^V$ | cost per vehicle |

The variables to determine are the following:

| | |
|---|---|
| $y_{j1,j2}$ | $= 1$ if job $J_{j1}$ is scheduled before job $J_{j2}$, 0 otherwise, $1 \leq j1, j2 \leq n$ |
| $Z_k$ | $= 1$ if vehicle $k$ is used, 0 otherwise, $1 \leq k \leq n$ |
| $z_{j,k}$ | $= 1$ if job $J_j$ departs on vehicle $k$, 0 otherwise, $1 \leq j \leq n$, $1 \leq k \leq n$ |
| $x_{j1,j2,k}$ | $= 1$ if job $J_{j1}$ and job $J_{j2}$ are transported in vehicle $k$ and $J_{j1}$ is delivered before $J_{j2}$, 0 otherwise, $1 \leq j1, j2 \leq n$, $1 \leq k \leq n$ |
| $D_j^M$ | estimation of the delivery due date of job $J_j$, $1 \leq j \leq n$ |
| $C_{i,j}$ | Completion time of job $J_j$ on machine $M_i$, $1 \leq j \leq n$, $1 \leq i \leq m$ |
| $F_k$ | departure time of vehicle $k$, $1 \leq k \leq n$ |
| $PT_j^M \geq 0$ | estimation of the tardiness of job $J_j$ for the manufacturer, $1 \leq j \leq n$ |
| $IC$ | total inventory costs |
| $PPC^M$ | pseudo penalty cost of the manufacturer |

The following variables will be known after the 3PL provider gives the delivery dates to the manufacturer.

| | |
|---|---|
| $D_j^{3PL}$ | delivery completion time of job $J_j$ |
| $T_j^M$ | tardiness of job $J_j$ |
| $VC$ | final vehicle cost |

### 2.4.1 Manufacturer

The scheduling problem of the manufacturer is solved by the following Mixed Integer Linear Programming model.

$$\text{Minimize } PTC^M = IC + PPC^M + VC \tag{11}$$

The relative order between two jobs $J_{j1}$ and $J_{j2}$ ($\forall j1, j2 \in \{1, ..., n\}, j1 \leq j2$) is given by the following constraints:

$$y_{j1,j2} + y_{j2,j1} = 1 \tag{12}$$

The resource constraints allow to define the completion time of a job on any machine $M_i$ ($\forall i \in \{1, ..., m\}$, $\forall j1, j2 \in \{1, ..., n\}$, $j1 \neq j2$):

$$C_{i,j2} \geq C_{i,j1} + p_{i,j2} - My_{j1,j2} \tag{13}$$

The routing constraints are given on any machine $M_i$ ($i \in \{2, ..., m\}$) and for any job $J_j$ ($j \in \{1, ..., n\}$):

$$C_{i,j} \geq C_{i-1,j} + p_{i,j} \tag{14}$$

Each job $J_j$ is transported in a vehicle ($\forall j \in \{1, ..., n\}$), therefore:

$$\sum_{k=1}^{n} z_{j,k} = 1 \tag{15}$$

Each vehicle $k$ ($\forall k \in \{1, ..., n\}$) leaves after the completion time of all jobs transported by this vehicle ($\forall j \in \{1, ..., n\}$):

$$F_k \geq C_{m,j} - M(1 - z_{j,k}) \tag{16}$$

A job cannot leave before its vehicle leaves ($\forall j \in \{1, ..., n\}$, $\forall k \in \{1, ..., n\}$):

$$f_j \geq F_k - M(1 - z_{j,k}) \tag{17}$$

A vehicle $k$ ($\forall k \in \{1, ..., n\}$) is used once it transports a job:

$$MZ_k \geq \sum_{j=1}^{n} z_{j,k} \tag{18}$$

In the same batch $k$ ($\forall k \in \{1, ..., n\}$), job $J_{j1}$ ($\forall j1 \in \{1, ..., n\}$) has a predecessor with a smaller due-date, or has the manufacturer site as predecessor:

$$\sum_{i \in \{0\} \cup \{j / d_j \leq d_{j1}\}} x_{i,j1,k} = z_{j1,k} \tag{19}$$

In the same batch $k$ ($\forall k \in \{1, ..., n\}$), job $J_{j1}$ ($\forall j1 \in \{1, ..., n\}$) has a successor with a greater due-date, or has the depot site as successor:

$$\sum_{i \in \{n+1\} \cup \{j / d_j \geq d_{j1}\})} x_{j1,i,k} = z_{j1,k} \tag{20}$$

The manufacturer site has a successor during the route of vehicle $k$ ($\forall k \in \{1, ..., n\}$) only if vehicle $k$ is used:

$$\sum_{j=1}^{n} x_{0,j,k} \leq Z_k \tag{21}$$

The estimation of the delivery date is given by the following constraints ($\forall j1, j2 \in \{1, ..., n\}$ and $\forall k \in \{1, ..., n\}$):

$$D_{j2}^M \geq D_{j1}^M + t_{j1,j2} - M(1 - x_{j1,j2,k}) \tag{22}$$

The following constraints give a lower bound of the delivery date of job $J_j$ ($\forall j \in \{1, ..., n\}$) transported in vehicle $k$ ($\forall k \in \{1, ..., n\}$):

$$D_j^M \geq F_k + t_{0,j} - M(1 - z_{j,k}) \tag{23}$$

The estimation of the tardiness for job $J_j$ ($\forall j \in \{1, ..., n\}$) is given by the following constraint:

$$PT_j^M \geq D_j^M - d_j \tag{24}$$

The costs are given in the following expressions:

$$PPC^M = \sum_{j=1}^{n} \pi_j^M PT_j^M \tag{25}$$

$$IC^{WIP} = \sum_{j=1}^{n} (C_{m,j} - C_{1,j}) q_j h_j^{WIP} \tag{26}$$

$$IC^{FIN} = \sum_{j=1}^{n} (\sum_{k=1}^{V} f_j - C_{m,j}) q_j h_j^{FIN} \tag{27}$$

$$IC = IC^{WIP} + IC^{FIN} \tag{28}$$

$$VC = c^V \sum_{k=1}^{n} Z_k \tag{29}$$

The objective function is:

$$\text{Minimize } IC + PPC^M + VC \tag{30}$$

This model contains $O(n^3)$ binary variables, $O(nm)$ continuous variables, $O(n^3)$ big-M constraints and $O(n^2 m)$ constraints.

### 2.4.2 3PL provider

We assume that the optimization of the routing for each of the vehicles is independent and can thus be optimized separately. Therefore, for each trip, the set of delivery dates $D_j^M$ is given by the manufacturer and we assume that vehicle $k$ has to leave the manufacturer site at time $F_k$, which is also given. Furthermore, we assume that a batch contains $s$ jobs to deliver.

The data required by the 3PL provider are the following:

| | |
|---|---|
| $s$ | number of sites to visit (sites $1, .., s$ are the customer sites, site 0 is the manufacturer site and site $s+1$ is the 3PL provider site) |
| $D_j^M$ | estimated delivery date for site $j$, $1 \leq j \leq s$ |
| $F_k$ | departure time from the manufacturer site of vehicle $k$, $1 \leq k \leq n$ |
| $t_{j1,j2}$ | travel time between site $j1$ and site $j2$, $0 \leq j1, j2 \leq s+1$ |
| $M$ | an arbitrary high value (can be set to $2 \times \sum_{j1} \sum_{j2} t_{j1,j2}$) |

The costs to be taken into account are the following.

| | |
|---|---|
| $c_{j1,j2}$ | cost of travel time between site $j1$ and site $j2$, $0 \leq j1, j2 \leq s+1$ |
| $\pi_j^{3PL}$ | penalty cost of 3PL provider for an excessive tardiness of job $J_j$, $1 \leq j \leq s$ |

The variables that have to be determined are the following.

| | |
|---|---|
| $x_{j1,j2}$ | $= 1$ if site $j1$ is visited just before site $j2$, 0 otherwise, $0 \leq j1, j2 \leq s+1$ |
| $D_j^{3PL}$ | date at which site $j$ is visited, i.e. delivery date of $J_j$, $1 \leq j \leq s$ |
| $T_j^{3PL} \geq 0$ | tardiness of delivery of $J_j$, $1 \leq j \leq s$ |
| $RC$ | routing cost |
| $PC^{3PL}$ | total penalty cost of 3PL provider |

The routing of the 3PL provider is determined by solving the following Mixed Integer Linear Programming model.

$$\text{Minimize } TC^{3PL} = RC + PC^{3PL} - VC \tag{31}$$

Any site $s_1$ ($\forall s_1 \in \{0, ..., s\}$) (excluding 3PL depot), has one successor site in $\{1, ..., s+1\}$.

$$\sum_{s2=1}^{s+1} x_{s1,s2} = 1 \tag{32}$$

Any site $s_1$ ($\forall s_1 \in \{1, ..., s+1\}$) (excluding manufacturer site), has one predecessor site in $\{0, ..., s\}$.

$$\sum_{s2=0}^{s} x_{s1,s2} = 1 \tag{33}$$

A lower bound of the delivery date of job $J_j$ ($\forall j \in \{1, ..., s\}$) is the following (remember that $F_k$ is a data here):

$$D_j^M \geq F_k + t_{0,j} \tag{34}$$

The arrival date of each job at the customer site is given by the following constraints ($\forall j1 \in \{1, ..., s\}$, $\forall j2 \in \{1, ..., s\}$):

$$D_{j2}^{3PL} \geq D_{j1}^{3PL} + t_{j1,j2} - M(1 - x_{j1,j2}) \tag{35}$$

The tardiness expression and the costs are given in the following expressions (remember that $D_j^M$ is a data here):

$$T_j^{3PL} \geq D_j^{3PL} - D_j^M, \quad \forall j \in 1, .., s \tag{36}$$

$$PC^{3PL} = \sum_{j=1}^{s} \pi_j^{3PL} T_j^{3PL} \tag{37}$$

$$RC = \sum_{j1=0}^{s} \sum_{j2=1}^{s+1} c_{j1,j2} x_{j1,j2} \tag{38}$$

This model contains $O(s^2)$ binary variables, $O(s)$ continuous variables, $O(s^2)$ big-M constraints and $O(s)$ constraints.

# 3 Heuristic resolution methods

Heuristic methods are developed to propose good solutions to the manufacturer problem. This is due to the complexity of the problem, which makes the exact resolution very difficult within reasonable computation time.

In this section, we only propose to solve the manufacturer problem, as each sub-problem of the 3PL provider is a VRP with soft due dates and few items which can be easily solved by the classical methods found in the literature (the MILP proposed in Section 2.4.2 is sufficient).

Three metaheuristic algorithms are proposed: a GRASP algorithm, a Tabu Search, and a Genetic Algorithm. These methods present some common features, which are first presented here: a fitness function, a two-phase heuristic (scheduling first, batching second) to build a good initial solution and local search protocols.

The complete algorithms will be presented in Section 3.5.

Note, that in the following, we consider to work on a two-machine production chain.

## 3.1 Coding of a solution

A solution to the problem is defined by the starting times of the jobs on the machines and a composition of batches to deliver. From this, it is possible to determine the inventory costs, the departure time of the batches for delivery, thus knowing their composition – and applying the EDD rule for delivery – the estimation of the delivery dates. So, all the elements are known to evaluate a solution.

A coding of a solution is a sequence of batches. The order in which the jobs get assigned to the batches is the same as the processing sequence on the machines. The order in which the jobs are delivered in each batch is given by EDD order. This coding will be used for the GRASP and the Tabu Search algorithms.

**Example:** We consider the following coding for a problem with $n = 8$ jobs.

$$\sigma = \{\{J_4, J_2, J_3\}, \{J_1, J_6\}, \{J_8, J_5, J_7\}\}$$

This coding corresponds to a production sequence equal to $(J_4, J_2, J_3, J_1, J_6, J_8, J_5, J_7)$, to the composition of three batches, the first one being composed by the jobs $J_4$, $J_2$ and $J_3$.

## 3.2 Fitness function

The fitness function is the evaluation method of a coded solution.

Let consider a coding $\sigma$. We denote by $\nu$ the number of batches in $\sigma$. This corresponds to the number of vehicles which are used. Notice that the cost $VC$ is a constant in this case $VC = c^V \nu$ and therefore it does not appear in the objective function. We assume w.l.o.g. that the jobs in $\sigma$ are numbered from $J_1$ to $J_n$, which makes the routing constraints (the order in which jobs visit the machines) easier to present.

We know which is the last job of each batch. We denote by $E_k$ the index of the last job in batch $k$, $1 \leq k \leq \nu$. We also know the assignment of jobs to batches and we denote by $B_j$ the batch assigned to job $J_j$, $1 \leq j \leq n$.

With a simple preprocessing, it is also possible to determine the time needed to deliver a job following the departure date of its vehicle. We denote by $TT_j$ the travel time to deliver $J_j$.

The optimal schedule of the jobs (starting times of jobs) can then be determine by solving the following Linear Programming model called $LP^{Fit}$. The data and variables have been previously defined before.

$$LP^{Fit} \tag{39}$$

$$\text{MIN } IC + PPC^M \tag{40}$$

$$\text{s.t. } C_{1,1} \geq p_{1,1} \tag{41}$$

$$C_{i,j} \geq C_{i,j-1} + p_{i,j} \qquad \forall j \in \{1, ..., n\}, \forall i \in \{1, ..., m\} \tag{42}$$

$$C_{i,j} \geq C_{i-1,j} + p_{i,j} \qquad \forall j \in \{1, ..., n\}, \forall i \in \{1, ..., m\} \tag{43}$$

$$F_k = C_{m,E_k} \qquad \forall k \in \{1, ..., \nu\} \tag{44}$$

$$f_j = F_{B_j} \qquad \forall j \in \{1, ..., n\} \tag{45}$$

$$D_j^M = f_j + TT_j \qquad \forall j \in \{1, ..., n\} \tag{46}$$

$$(24), (25), (26), (27), (28) \tag{47}$$

This LP model allows to optimally evaluate a coding of a solution.

In the case of $m = 2$ machines and under some hypotheses, it is possible to propose a more efficient method.

We assume that the penalty costs to the customers are much more important than the inventory costs. Therefore, we will not delay the departure of a vehicle in order to possibly save an inventory cost. The

consequence is that we respect the completion time of the jobs on the last machine (i.e. the vehicles departure time) in a left-shifted schedule.

The critical path of the batch is the set of operations which give the duration of the batch. In order to minimize the inventory costs, we consider three types of jobs in each batch, depending on their relative position to the critical path.

- type 1: the jobs composing the critical path: we cannot move such a job without modifying the departure date of the vehicle. So we cannot modify the inventory cost of these jobs *(dark gray fill in Fig. 1)*.
- type 2: the jobs on the first machine which do not belong to the critical path: we right-shift these jobs, in order to minimize their Work In Process inventory cost *(light gray fill in Fig.1)*.
- type 3: the jobs on the second machine which do not belong to the critical path: a right-shift movement on such a job increases the WIP inventory cost but decreases the Final inventory cost. We schedule these jobs with an ad-hoc algorithm *(white fill in Fig.1)*.

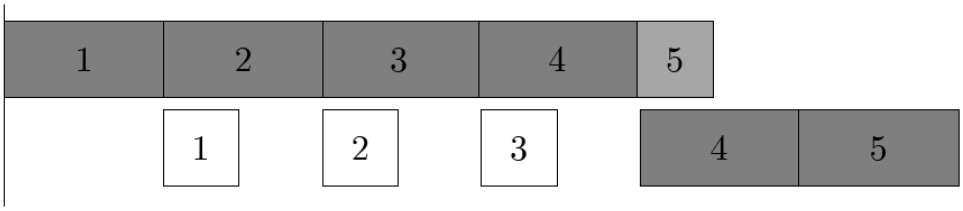



Figure 1: Three different kinds of jobs

We denote by $h_j^\delta$ the difference of $h_j^{WIP} - h_j^{FIN}$ for the job $J_j$ of type 3. If $h_j^\delta > 0$, a right-shift of $J_j$ decreases its inventory cost. We assume w.l.o.g. that the jobs of type 3 are numbered from $J_1$ for the first one to $J_o$ for the last one. Algorithm 2 allows to define the position of the jobs of type 3 on machine $M_2$.

**Example:** In Fig.1, we have $h_1^\delta = 3$, $h_2^\delta = -2$ and $h_3^\delta = -2$. Job 1 is right shifted because $h_1^\delta > 0$. Then, jobs 1 and 2 are right shifted because $h_1^\delta + h_2^\delta = 1 > 0$. Finally, because $h_1^\delta + h_2^\delta + h_3^\delta = -1 < 0$, we do not shift again these jobs to the right. Applying Alg. 2 leads to the solution illustrated in Fig. 2.

The $IC$ is evaluated from this last configuration.

### 3.3 Initial solution

The methods start from an initial solution. To generate an initial solution, we determine first the sequencing of jobs and then we choose the composition of vehicles (or batching). Of course, because this method uses random parameters, it is not guaranted that the methods start with the same initial solution.

**Algorithm 2** Min inventory cost algorithm

$B \leftarrow \emptyset$, $Bh^{\delta} \leftarrow 0$
**for** $j \in (1, \ldots, o)$ **do**
  $B \leftarrow B \cup \{J_j\}$
  $Bh^{\delta} \leftarrow Bh^{\delta} + h_j^{\delta}$
  **if** $Bh^{\delta} > 0$ **then**
    *All the jobs in $B$ are right-shifted*
  **else**
    $B \leftarrow \emptyset$, $Bh^{\delta} \leftarrow 0$
  **end if**
**end for**

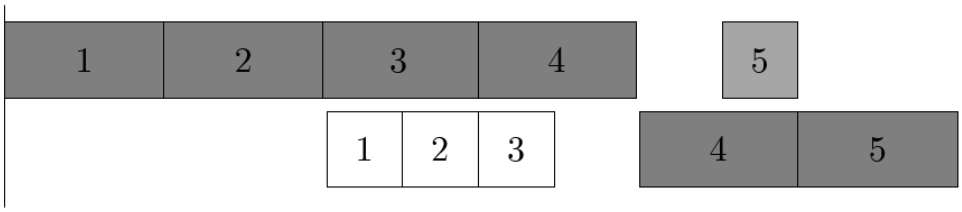



Figure 2: Result of the min inventory cost procedure

### 3.3.1 Sequencing of jobs

To build the sequence, we choose the first job randomly among a restricted list based on the job's due dates. A *restricted list* is composed by the jobs with a due date belonging to a certain interval. The size of this interval is related to a parameter $\lambda$. The details of the algorithm are given in Alg. 3.

**Algorithm 3** *sequencing_heuristic* algorithm

1: Parameters $\lambda$,
2: $\sigma = \emptyset$, $\mathcal{J} = \{J_j, j \in \{1, ..., n\}\}$
3: **while** $\mathcal{J} \neq \emptyset$ **do**
4: $d^{min} = \min_{J_j \in \mathcal{J}} d_j$
5: $d^{max} = \max_{J_j \in \mathcal{J}} d_j$
6: List $= \{J_j / d^{min} \leq d_j \leq d^{min} + \lambda \times (d^{max} - d^{min})\}$
7: Select randomly $J_j$ in List
8: $\sigma \leftarrow \sigma + J_j$
9: $\mathcal{J} \leftarrow \mathcal{J} \setminus \{J_j\}$
10: **end while**
11: **return** $(\sigma)$

Notice that we can generate a solution starting by the end of the schedule (changing line 6 in Alg. 3 to List $= \{J_j / d^{max} - \lambda \times (d^{max} - d^{min}) \leq d_j \leq d^{max}\}$ and line 8 to $\sigma \leftarrow J_j + \sigma$).

### 3.3.2 Batching of the sequence

We develop three methods for dispatching the jobs into batches: the two first ones are based on the cost of adding a job to an existing batch. The third one is inspired by the Split algorithm of Prins [Prins, 2004].

**First batching method**: From the schedule, we build batches by evaluating the additional cost of a new job in the current batch (denoted $Cost_1$) in comparison to the additional cost of creating a new batch (denoted $Cost_2$). If creating a new batch (and hiring a new vehicle) is cheaper than adding the job to the current batch, this option is chosen. Otherwise, we refer to the parameter DETERMINIST of the algorithm. If this parameter is true, the new job is assigned to the current batch, otherwise the decision is taken randomly, depending on the ratio between $Cost_1$ and $Cost_2$ (see Alg. 4).

**Split algorithm**: The Split algorithm [Prins, 2004] is presented for the $DVRP$ (*Distance-constraint Vehicle Routing Problem*) in order to determine an optimal set of trips for customers, respecting an already known sequencing order. The principle is to depict the problem by an oriented graph representing all the possible trips, with the associated costs on edges. The shortest path in this graph gives the optimal set of travels and therefore, the batching.

**Algorithm 4** *batching_heuristic* algorithm

Parametesr: $\sigma$ *(from GRASP heuristic)*, DETERMINIST
Initialization: $\mathcal{B} = \emptyset$ // set of batches
$B = \{\sigma[1]\}$ // current batch
$k = 2$
**while** $k \leq n$ **do**
  $J_j = \sigma[k]$
  $Cost_1$ = cost of assignment of $J_j$ to batch $B$
  $Cost_2$ = cost of assignment of $J_j$ to a new batch
  **if** $Cost_1 > Cost_2$ **then**
    in_current_batch = FALSE
  **else if** DETERMINIST **then**
    in_current_batch = TRUE
  **else**
    in_current_batch = $\frac{Cost_1}{Cost_2} \leq rand(0,1)$
  **end if**
  **if** in_current_batch **then**
    $B = B \cup \{J_j\}$ // $J_j$ is put in current batch $B$
  **else**
    $\mathcal{B} = \mathcal{B} \cup \{B\}$ // batch $B$ is added to $\mathcal{B}$
    $B = \{J_j\}$ // a new batch is created
  **end if**
  $k = k + 1$
**end while**
$\mathcal{B} = \mathcal{B} \cup \{B\}$
**return** $\mathcal{B}$

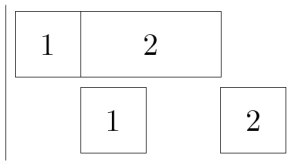



Figure 3: Jobs are left-shifted

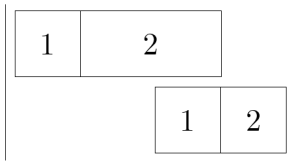
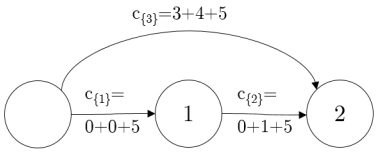



Figure 4: Minimizing inventory




$$c_X = IC_X + PPC_X^M + c^V$$

Figure 5: Batching cost graph

In our case, the input of the algorithm is a sequence of jobs. We define a graph $G = (V, A)$ where $V$ contains $n + 1$ vertices indexed from 0 to $n$, $A$ contains one arc for each pair $(i, j)$, with $i < j$, representing the batch containing the $(i + 1)^{th}$ to the $j^{th}$ job of the sequence. The cost of this edge represents the cost associated to this batch: (1) the inventory cost plus (2) the estimation of tardiness penalties plus (3) the cost of one vehicle. This evaluation can be obtained by using $LP^{Fit}$ with only one batch, release dates on the machines ($C_{i,1} \geq R_i$, $\forall i \in \{1, ..., m\}$), and jobs of type 3. In the case of $m = 2$ machines, $IC$ is given by Alg. 2 for the jobs of type 3, $PPC^M$ is easily computed and $VC$ is equal to $c^V$.

After building this graph, we use Bellman's algorithm to find the shortest path. This evaluation is reasonably fast because $G$ has no cycle. The worst case complexity is $O(n^2)$.

**Example:** Let consider a two-job instance with $p_{11} = p_{12} = p_{22} = 1$, $p_{21} = 2$, $h_1^{WIP} = h_2^{WIP} = 1$, $h_1^{FIN} = h_2^{FIN} = 2$ and $c^V = 5$. The job sequence is $(J_1, J_2)$, the associate graph contains 3 vertices and 3 edges for the 3 batches $\mathcal{B}_1 = \{J_1\}$, $\mathcal{B}_2 = \{J_2\}$ and $\mathcal{B}_3 = \{J_1, J_2\}$. The jobs are left shifted to have the departure date of the batch (see Fig. 3).

The departure time of the batch $\mathcal{B}_1$ finishing by job $J_1$ is 2 and the departure time of batches $\mathcal{B}_2$ and $\mathcal{B}_3$ finishing by job $J_2$ is 4. If we denote by $c_X$ the cost associated to batch $\mathcal{B}_X$, we have $c_X = IC_X + PPC_X^M + c^V$. $IC$ for batches $\mathcal{B}_1$ and $\mathcal{B}_2$ are equals to 0: $IC_1 = IC_2 = 0$. $IC$ for batch $\mathcal{B}_3$ is equal to $IC_3 = h_1^{WIP} \times 1 + h_1^{FIN} \times 1 = 3$.

We assume that the data for computing $PPC^M$ leads to the following results (not detailed here for sake of simplicity): $PPC_1^M = 0$, $PPC_2^M = 1$ and $PPC_3^M = 4$.

Using Bellman on the graph of Fig. 5, we obtain the shortest path equal to $\{(0, 1), (1, 2)\}$ with an associate cost of 11. It means that $J_1$ and $J_2$ are in two separate batches. If $c^V$ increases from 5 to 7, the shortest path changes to $\{(0, 2)\}$ with an associated cost of 14.

## 3.4 Local Search

We use local search in order to improve the current solution quickly.

### 3.4.1 Operator

We present the neighborhood operators in this section.

**Swap**: Exchange two jobs in the sequence. To avoid symmetry, let $i$ and $j$ be the positions of two jobs, these jobs can be swapped if and only if $i < j$. For example :

$$\text{Swap of 2 and 5: } \{\{1,\underline{2},3,4\}\{\underline{5},6,7\}\} \rightarrow \{\{1,\underline{5},3,4\}\{\underline{2},6,7\}\}$$

**EBSR/EFSR - Extract and Backward / Forward Shift Reinsertion**: This operator extracts a job $J_i$ from position $k$ and reinsert it in the sequence at a position $h < k$ (backward), or at a position $h > k$ (forward). We denote by $J_j$ the job which was at position $h$. Job $J_i$ is inserted in the batch of $J_j$.

$$\text{EFSR of } J_2 \text{ at position 5: } \{\{1,\underline{2},3,4\}\{5,6,7\}\} \rightarrow \{\{1,3,4\}\{5,\underline{2},6,7\}\}$$

$$\text{EBSR of } J_5 \text{ at position 2: } \{\{1,2,3,4\}\{\underline{5},6,7\}\} \rightarrow \{\{1,\underline{5},2,3,4\}\{6,7\}\}$$

This operator could lead to an empty batch if the job which is extracted was the only job of its batch. The two operators are combined in the EBFSR operator by testing for a job $J_i$ a reinsertion at a position $h$ smaller than $k$, and then greater than $k$.

**Merging**: This operator merges two batches:

$$\text{Merging of } \mathcal{B}_1 \text{ and } \mathcal{B}_2\text{: } \{\underline{\{1,2,3\}\{4,5\}}\{6,7\}\} \rightarrow \{\underline{\{1,2,3,4,5\}}\{6,7\}\}$$

**Division**: This operator splits a batch in two batches just after a target job.

$$\text{Division of } \mathcal{B}_1 \text{ after job } J_3\text{: } \{\{1,2,\underline{3},4,5\}\{6,7\}\} \rightarrow \{\{1,2,3\}\{4,5\}\{6,7\}\}$$

This operator is the only one which can increase the number of batches.

### 3.4.2 Neighborhood exploration

To avoid an explosion of computation time associated to our local search, the operators are limited by a parameter $\Delta$ (depending of the instance size) which reduces the distance of the jobs (in terms of positions) considered by the operators *Swap* and *EBFSR*.

**Random start for exploration** The natural way for exploring the neighborhood associated to an operator is to fix the position of the target job to a minimal value and to proceed to the neighborhood exploration by increasing the index up to its maximum value.

In case of using a First Improvement for local search, the first part of a solution will be improved first. The problem is that the operator will continue searching improvements in this first part at each step of the local search on this solution, wasting time on the evaluation as this part will be difficult to improve.

| | |
|---|---|
| Begin : | 2 4 |
| 1 2 | 3 4 |
| 1 3 | 3 5 |
| 2 3 | 4 5 |
| 2 4 | End: |
| 3 4 | Begin : |
| 3 5 | 1 2 |
| 4 5 | 1 3 |
| End: | 2 3 |
| Natural exploration | Random start exploration |

Figure 6: Exploration for operator *Swap*, $\Delta = 2$, $n = 5$

A random start is implemented for each operator, in order to explore from this point rather than from the beginning. This is illustrated in Fig. 6.

The different operators are always used in the following order: *Swap* - *EBFSR* - *Merging* - *Division*.

#### 3.4.3 First or Less Worst Improvement - FLWI

Based on these operators we develop the following descent algorithm. We explore the neighborhood and repeat the process with the first solution which improves the current solution. If such a solution does not exist, the method repeat the process with the best neighbor (without improving the current solution) as described in Alg.5. In this algorithm, we denote the current solution by $S$, the neighbor by $S'$ and the best neighbor by $S^*$. *random_start* is a function defining the starting point in the neighborhood operator. *apply_mouv* creates the neighbor of $S$.

### 3.5 Methods description

With these different components, we build three different algorithms for solving this problem: a GRASP algorithm, a Tabu Search algorithm and a Genetic Algorithm.

#### 3.5.1 GRASP

First, we propose a GRASP (Greedy Randomized Adaptive Search Procedure) algorithm. The idea of this algorithm is to quickly generate a considerable number of good quality solutions and to return the best one. Solutions are generated by using the previous sequencing and batching procedures (Alg. 3 with parameter $\lambda \in \{0, 1\}$, SPL_PARAM $\in$ {DETERMINIST, NON-DETERMINIST, SPLIT}) and the solution is improved by FLWI local search until a local minimum is found (see Alg. 6).

**Algorithm 5** *Step_FLWI*

1: Input: $S$
2: $Tab\_Ope = [Swap, EBFSR, Merging, Division]$
3: $f(S') = \infty$
4: **for each** Operator *ope* in $Tab\_Ope$ **do**
5: $ope.random\_start()$
6: **while** $ope.have\_next\_mouv()$ **do**
7: $S' = ope.apply\_mouv(S)$
8: **if** $f(S') < f(S)$ **then**
9: **return** $S'$
10: **end if**
11: **if** $f(S') < f(S^*)$ **then**
12: $S^* = S$
13: **end if**
14: **end while**
15: **end for**
16: **return** $S^*$

### 3.5.2 Tabu Search algorithm

In order to improve this method, we extend the local search mechanism of FLWI with a tabu list. Proposed by [Glover, 1989], the principle is to keep a list of the different operators and moves which allowed to improve the current solution. All moves in this list are considered as tabu and cannot be reversed even in order to improve the solution. This mechanism has been developed to escape from local minima and to continue the search. The procedure called *Step_FLWI*-Tabu line (11 in Alg. 7) is similar to Alg. 5, except that line 7 is changed to test if neighbor $S'$ belongs to the Tabu list or not. The neighbor is considered if it does not belong to the list.

The method has three parameters. The first is the number of iterations allowed without improvement of the current best solution, called NB_MAX_W_IMPR. When this parameter is reached, the algorithm uses a diversification procedure to explore another part of the landscape: we apply a certain number of random moves on the best solution found during the last descent. The second parameter indicates which neighborhood operator is used among SWAP and EBFSR. Finally, the third parameter is a coefficient $\gamma$ and the number of moves is equal to the number of jobs multiplied by $\gamma$.

**Algorithm 6** GRASP heuristic

Parameters $\lambda$, SPL_PARAM, $CPUmax$

$f(S^*) = \infty$

**while** $CPU \leq CPUmax$ **do**

  *//Initial solution creation*

  $\delta = sequencing\_heuristic(\lambda)$ (Alg. 3)

  $S = batching\_heuristic(\delta, \text{SPL\_PARAM})$ (Alg. 4 or Split)

  *//First Improvement*

  $S' = Step\_FLWI(S)$

  **while** $f(S') \leq f(S)$ **do**

    $S = S'$, $S' = Step\_FLWI(S)$

  **end while**

  *//Update best solution*

  **if** $f(S) \leq f(S^*)$ **then**

    $S^* = S$

  **end if**

**end while**

**return** $S^*$

**Algorithm 7** Tabu heuristic

Parameters $\lambda$, SPL_PARAM, TABU_S, NB_MAX_W_IMPR, $CPUmax$

*//Initial solution creation*

$\delta = sequencing\_heuristic(\lambda)$

$S = batching\_heuristic(\delta, \text{SPL\_PARAM})$

$f(S^*) = \infty$

**while** $CPU \leq CPUmax$ **do**

  *//Improvement*

  $nb\_ite = 0$

  $S_1 = S$

  **while** $nb\_ite <$ NB_MAX_W_IMPR **do**

    $S_2 = Step\_FLWI\text{-Tabu}(S_1)$

    $Update\_tabu\_List(mouv(S_1, S_2))$

    $S_1 = S_2$

    **if** $f(S_1) < f(S)$ **then**

      $S = S_1$, $nb\_ite = 0$

    **else**

      $nb\_ite++$

    **end if**

  **end while**

  *// Update best solution*

  **if** $f(S) < f(S^*)$ **then**

    $S^* = S$

  **end if**

  $Reset\_Tabu\_List()$

  $S = Diversification(S)$

**end while**

**return** $S^*$

### 3.5.3 Genetic algorithm

A Genetic Algorithm manages a population of solutions using rules inspired by the natural evolution law as reproduction, competition and selection, in order to adapt and improve over the generations. In our case, this algorithm depends on parameters $\delta_{pop}$ (size of the population) and $\alpha$ (mutation probability).

**Genotype** Each solution in the population is unique and represented by its own gene. This gene is the sequence of jobs on the manufacturer's machines. To obtain the value of the objective function, we first apply the procedure Split and then the fitness function. The value is stored with the solution.

**Initial population** The initial population is built with the sequencing and batching procedures presented below. We use parameter $\lambda = 0.3$ and SPLIT to generate the whole population. A small part of the population ($\alpha\%$) is improved with FLWI local search procedure.

**Selection** A new solution (an offspring) is the combination of two individuals. Each parent is chosen by using a binary tournament: two solutions are picked at random from the population and the one with the best fitness value is conserved.

**Crossover** The crossover is the procedure defining an offspring from two parents. We use the LOX operator for Linear Order crossover. The principle is to keep the central part of the first parent (delimited by two random indices) and to move the other elements backward and forward, based on the sequence of the second parent, in order to preserve a part of the supposedly good solution as well as the relative order of the other jobs.

An example presented in Tab.1 illustrates the crossover.

**Mutation** Each offspring is subject to a mutation with probability $\alpha$. In this case, the local search is applied until a local optima is found.

**Population replacement** An offspring is added to the current population only if its gene is not already present in the population. We consider that two genes are identical if they have the same objective function value, thus all the objective function values in the population are different. If an offspring is added, it takes the place of an element present in the half of the population with the worst objective function value. In order to process this operation easily, the population is always sorted.

This loop is applied in the population until the time constraint is reached. The first element of the population is returned.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Parent 1 : | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Parent 2 : | 5 | 6 | 4 | 1 | 2 | 3 | 7 |

Random indices: $c_1 = 2,\ c_2 = 5$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Parent 1 : | 1 | 2 | $\lvert\underline{3}$ | $\underline{4}$ | $\underline{5}\rvert$ | 6 | 7 |
| Parent 2 : | $\underline{5}$ | 6 | $\underline{4}$ | 1 | 2 | $\underline{3}$ | 7 |
| Offspring: | - | - | 3 | 4 | 5 | - | - |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Parent 2 : | X | 6 | X | 1 | 2 | X | 7 |
| Offspring: | 6 | 1 | X | X | X | 2 | 7 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Offspring: | 6 | 1 | 3 | 4 | 5 | 2 | 7 |

Table 1: LOX crossover example

# 4 Computational results

We present in this section the computational resuls which have been performed on a machine with an Intel Core i7-7820HQ and 16,00 Go RAM.

## 4.1 Data generation

The results presented in this section have been obtained with instances generated according to the following specifications.

We consider a 2-machine flow shop problem.

We set a parameter $SSQ = 100$. For each job $J_j$ and each machine $M_i$, the processing time $p_{i,j}$ is randomly chosen between 1 and $SSQ$. The different sites are placed randomly on a square of size $3SSQ \times 3SSQ$ and the distance $t_{j1,j2}$ between sites $j1$ and $j2$ is the classical euclidian distance.

Costs are fixed in order to obtain an optimal solution where the number of vehicles is not equal to 1 and not equal to $n$. For each job $J_j$ the quantity $q_j$ is equal to 100, the work-in-process inventory cost $h_j^{WIP}$ is equal to 1, the finished product inventory cost $h_j^{FIN}$ is equal to 2, the manufacturer penalty cost $\pi^M$ is equal to 2 and the 3PL penalty cost $\pi^{3PL}$ is equal to 9. The cost of vehicle $c^V$ is 200000.

Sets of 10 instances are generated for each value of $n \in \{6, 7, 8, 9, 10, 20, 30, 40, 50, 60, 100\}$. Instances with $n \in \{6, 7, 8, 9\}$ are used for testing the MILP model and instances with $n \in \{10, 20, 30, 40, 50, 60, 100\}$ are used for testing the heuristic algorithms.

## 4.2 Results of the MILP

The solver used is CPLEX 12.7.1. The maximum computation time is limited to 30 minutes. Table 4.2 gives the results of the MILP.

| $n$ | Percentage of instances solved | Average time CPLEX (s) |
|---|---|---|
| 6 | 100% | <1 |
| 7 | 100% | 14 |
| 8 | 100% | 169 |
| 9 | 0% | – |

Table 2: MILP efficiency

We can notice that the computation time of CPLEX quickly increases with $n$. No instance with 9 jobs is solved before the time limit.

Notice also that the MILP for the 3PL provider is very efficient and can solve all the instances in less than one second (each batch contains very few jobs).

## 4.3 Metaheuristic algorithms

the proposed heuristics are based only on the first part of the model, in which the manufacturer chooses the schedule of his jobs and the batch composition. As a reminder, the objective function of this model is the sum of the inventory cost ($IC$) plus the vehicle cost ($VC$) plus the pseudo penalty cost ($PPC^M$).

### 4.3.1 Iterated initial algorithm

We test the algorithm which returns an initial solution with parameter $\lambda \in \{0.0, 0.3, 0.5\}$ and the three batching protocols denoted by {False, True, Split}, corresponding to the Non deterministic case, to the Deterministic case and to the Split algorithm respectively. In order to evaluate these different parameters, some preliminary experiments have been conducted. The nine configurations have been tested by running the algorithm without local search improvement and with 2000 runs per instance. The result given in Tab. 3 represent the number of times a configuration is better than the others.

We observe that the sequencing technique is better with parameter $\lambda \neq 0.0$ and for the Split algorithm as a batching technique.

| | $\lambda = 0.0$ | | | $\lambda = 0.3$ | | | $\lambda = 0.5$ | | |
|---|---|---|---|---|---|---|---|---|---|
| n | False | True | Split | False | True | Split | False | True | Split |
| 10 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 5 |
| 20 | 0 | 0 | 0 | 1 | 0 | 6 | 0 | 0 | 3 |
| 30 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 5 |
| 40 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 6 |
| 50 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 6 |
| 60 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 5 |
| 100 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 4 |
| Total | 0 | 0 | 0 | 1 | 0 | 34 | 0 | 1 | 37 |

Table 3: Evaluation of the iterated initial algorithm

#### 4.3.2 GRASP algorithm

The time limit in seconds has been fixed to $n/2$. We use the same sets of parameters as presented before. Tab. 4 presents the number of best solutions found for each parameter value.

| n | $\lambda = 0.0$ | | | $\lambda = 0.3$ | | | $\lambda = 0.5$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | False | True | Split | False | True | Split | False | True | Split |
| 10 | 8 | 8 | 3 | 8 | 9 | 9 | 9 | **10** | 9 |
| 20 | 0 | 4 | 1 | 1 | 2 | 1 | 2 | **7** | 2 |
| 30 | 1 | **3** | 2 | 0 | 2 | 1 | 0 | 0 | 1 |
| 40 | 0 | 2 | 1 | 1 | **4** | 0 | 0 | 2 | 0 |
| 50 | 0 | 3 | 0 | 0 | 1 | **5** | 0 | 0 | 1 |
| 60 | 0 | 2 | 1 | 0 | 0 | 2 | 0 | **3** | 0 |
| 100 | 0 | **4** | 2 | 0 | 0 | 2 | 0 | 2 | 0 |
| Total | 9 | **26** | 10 | 10 | 18 | 20 | 11 | 24 | 13 |

Table 4: Results of the GRASP algorithm

The results show that the Non Deterministic leads to weak performances, whatever the value of $\lambda$. The best configurations are obtained for $\lambda \in \{0.0, 0.5\}$ and for a Deterministic batching procedure (parameter equal to True). Good results are also obtained for $\lambda = 0.3$ and the use of the Split algorithm.

We consider that the best configuration is obtained with $\lambda = 0.0$ and parameter True.

Tab. 5 shows the average deviation from the best known solution returned by the iterated initial algorithm. This improvement, around 10%, is due to the application of the Local Search.

| n | $\lambda = 0.0$ | | | $\lambda = 0.3$ | | | $\lambda = 0.5$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | False | True | Split | False | True | Split | False | True | Split |
| 10 | 2.6 % | 2.5 % | -2.3 % | 2.6 % | 2.7 % | 2.7 % | 2.7 % | **2.8 %** | 2.7 % |
| 20 | 7.6 % | 8.4 % | 6.3 % | 7.6 % | 8.5 % | 7.8 % | 7.4 % | **8.7 %** | 7.9 % |
| 30 | 10.0 % | 10.5 % | 9.9 % | 9.0 % | 10.7 % | **10.8 %** | 9.0 % | 10.4 % | 10.5 % |
| 40 | 9.7 % | 10.8 % | 9.4 % | 8.7 % | **10.9 %** | 9.6 % | 8.9 % | 10.7 % | 9.7 % |
| 50 | 9.0 % | 11.2 % | 9.0 % | 7.5 % | 10.7 % | **11.5 %** | 9.1 % | 10.9 % | **11.5 %** |
| 60 | 9.7 % | 11.8 % | 10.8 % | 7.8 % | 11.5 % | 11.6 % | 8.9 % | **11.9 %** | 11.7 % |
| 100 | 7.9 % | **12.0 %** | 11.9 % | 7.0 % | 11.3 % | 11.9 % | 7.8 % | 11.6 % | 11.7 % |

Table 5: GRASP: average deviation from the iterated initial algorithm

#### 4.3.3 Tabu Search algorithm

In Tab. 6, we present the results of the Tabu Search algorithm with the following parameters. The initial solution is generated with the parameters $\lambda = 0.3$ and Split. The size of the tabu list is set at 7. Depending on the parameter nb_max_w_impr, we may accept 6, 10 or 12 iterations without improvement during a local search, before using the diversification procedure. The diversification is done with the SWAP or EBFSR operator and the number of random moves is proportional to the instance size and defined by the $\lambda$ parameter. The table gives the number of times the algorithm with the given configuration finds the best solution.

| $n$ | nb_max_w_impr = 6 | | | | | | nb_max_w_impr = 10 | | | | | | nb_max_w_impr = 12 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SWAP | | | EBFSR | | | SWAP | | | EBFSR | | | SWAP | | | EBFSR | | |
| $\gamma$ | 0.1 | 0.2 | 0.3 | 0.1 | 0.2 | 0.3 | 0.1 | 0.2 | 0.3 | 0.1 | 0.2 | 0.3 | 0.1 | 0.2 | 0.3 | 0.1 | 0.2 | 0.3 |
| 10 | 8 | **9** | **9** | 8 | 8 | 8 | 8 | 8 | 8 | **9** | 8 | 7 | 8 | 8 | 8 | 8 | 8 | 8 |
| 20 | 3 | **5** | 3 | 2 | 1 | 1 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 3 | 1 | 1 | 0 |
| 30 | **2** | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | **2** | 1 | 0 | 1 | 1 |
| 40 | 0 | 1 | 0 | **3** | 0 | 2 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 50 | 0 | 0 | 0 | 0 | 1 | 1 | **2** | 1 | 1 | 1 | **2** | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 60 | 2 | **3** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 100 | 1 | 1 | 1 | 0 | **3** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 |
| Total | 16 | **19** | 14 | 13 | 12 | 12 | 14 | 13 | 13 | 14 | 14 | 10 | 11 | 12 | 13 | 11 | 11 | 10 |

Table 6: Results of the Tabu Search algorithm

We can notice that no parameter really leads to a considerably better performance. It seems that the three first columns are better than the others and the three last are weaker, but the difference is not really significant.

Nevertheless, we consider that the configuration NB_MAX_W_IMPR = 6, SWAP and $\gamma = 0.2$ leads to the best results.

#### 4.3.4 Genetic Algorithm

In Tab. 7, we give the results of the Genetic Algorithm with the different population sizes $\delta_{pop} \in \{20, 30, 50\}$ and the probability of mutation $\alpha \in \{0.1, 0.2, 0.3\}$. The table gives the number of times the algorithm with the given configuration finds the best solution.

| $n$ | $\delta_{pop} = 20$ | | | $\delta_{pop} = 30$ | | | $\delta_{pop} = 50$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\alpha$=0.1 | $\alpha$=0.2 | $\alpha$=0.3 | $\alpha$=0.1 | $\alpha$=0.2 | $\alpha$=0.3 | $\alpha$=0.1 | $\alpha$=0.2 | $\alpha$=0.3 |
| 10 | 8 | 7 | 9 | 9 | 9 | **10** | 8 | **10** | **10** |
| 20 | 2 | 4 | 5 | 4 | 6 | 6 | 7 | **8** | 7 |
| 30 | 2 | 0 | 2 | 1 | **4** | 1 | 2 | **4** | 0 |
| 40 | **4** | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 50 | **3** | **3** | 2 | 0 | 0 | 0 | 1 | 1 | 0 |
| 60 | 2 | **4** | 1 | 2 | 1 | 0 | 0 | 0 | 0 |
| 100 | 2 | 0 | 1 | 1 | 1 | 0 | 0 | **3** | 2 |
| Total | 23 | 19 | 21 | 21 | 21 | 18 | 21 | **25** | 20 |

Table 7: Genetic algorithm : Number of best solution for each method

The performances of the GA are similar whatever the parameter settings. For small size instances (up to 30 jobs), the best parameters are $\delta_{pop} = 50$ and $\alpha = 0.2$, but for larger instances, the best parameters are $\delta_{pop} = 20$ and $\alpha = 0.2$. We consider that the later configuration is the best.

### 4.4 Comparison of the metaheuristic algorithms

We compare now the best configuration of the metaheuristic algorithms. Each algorithm is considered with the best set of parameters found in the previous experimentation and summarized bellow:

**GRASP:** $\lambda = 0.0$, TRUE

**Tabu Search:** NB_MAX_W_IMPR = 6, SWAP, $\gamma = 0.2$

**Genetic Algorithm:** $\delta_{pop} = 50$, $\alpha = 0.2$

#### 4.4.1 Global comparison

Table 8 indicates the number of times the method returns the best solution (#b) and the average relative deviation from this solution ($\Delta$).

| | GRASP | | Tabu | | Genetic | |
|---|---|---|---|---|---|---|
| $n$ | #b | $\Delta$ | #b | $\Delta$ | #b | $\Delta$ |
| 10 | 8 | 0.03% | 9 | 0.1% | **10** | 0 |
| 20 | 4 | 0.07% | 2 | 0.4% | **9** | 0.1% |
| 30 | 0 | 1.1% | 0 | 0.6% | **10** | 0.0% |
| 40 | 0 | 1.1% | 4 | 0.9% | **6** | 0.1% |
| 50 | 1 | 1.0% | **5** | 0.7% | 4 | 0.2% |
| 60 | 2 | 1.3% | **6** | 0.6% | 2 | 0.8% |
| 100 | 3 | 0.6% | 3 | 0.4% | **4** | 0.5% |
| Total/Aver. | 18 | 0.9% | 25 | 0.5% | **45** | 0.2% |

Table 8: Number of best solutions and average deviation for each method

We observe that the the Genetic Algorithm generally performs best. However, we can see, that, while the Genetic Algorithm is the best method for instances with up to 40 jobs, the Tabu Search method performs very well for instances with more than 50 jobs.

#### 4.4.2 Two by two comparison

Now we compare the methods two by two. For each couple and each set of instance, we indicate the number of times the method is better (#b) and the average deviation $\Delta$ defined by:

$$\Delta = \frac{M - O}{M}$$

with $M$ being the value obtained by the method and $O$ the value obtained by the other method.

| | GRASP | | TS | |
|---|---|---|---|---|
| $n$ | #b | $\Delta$ | #b | $\Delta$ |
| 10 | 8 | $-0.2\%$ | 10 | 0.2% |
| 20 | 6 | $-0.3\%$ | 6 | 0.3% |
| 30 | 3 | $-0.4\%$ | 7 | 0.4% |
| 40 | 3 | $-0.2\%$ | 7 | 0.2% |
| 50 | 2 | $-0.3\%$ | 8 | 0.3% |
| 60 | 3 | $-0.8\%$ | 7 | 0.7% |
| 100 | 4 | $-0.1\%$ | 6 | 0.1% |
| T. | 29 | -0,4% | 51 | 0,3% |

Table 9: GRASP and TS

| | GRASP | | GA | |
|---|---|---|---|---|
| $n$ | #b | $\Delta$ | #b | $\Delta$ |
| 10 | 8 | $-0.3\%$ | 10 | 0.3% |
| 20 | 4 | $-0.6\%$ | 9 | 0.6% |
| 30 | 0 | $-1.1\%$ | 10 | 1.1% |
| 40 | 0 | $-1.0\%$ | 10 | 1.0% |
| 50 | 3 | $-0.8\%$ | 7 | 0.8% |
| 60 | 4 | $-0.5\%$ | 6 | 0.5% |
| 100 | 5 | $-0.1\%$ | 5 | 0.1% |
| T. | 24 | $-0,6\%$ | 57 | 0,6% |

Table 10: GRASP and Genetic

| | TS | | GA | |
|---|---|---|---|---|
| $n$ | #b | $\Delta$ | #b | $\Delta$ |
| 10 | 9 | $-0,1\%$ | 10 | 0,1% |
| 20 | 3 | $-0,3\%$ | 9 | 0,3% |
| 30 | 0 | $-0,6\%$ | 10 | 0,6% |
| 40 | 4 | $-0,8\%$ | 6 | 0,8% |
| 50 | 5 | $-0,5\%$ | 5 | 0,4% |
| 60 | 8 | $0,2\%$ | 2 | $-0,3\%$ |
| 100 | 3 | $0,0\%$ | 7 | 0,0% |
| T. | 32 | $-0,3\%$ | 49 | 0,3% |

Table 11: Tabu and Genetic

These tables confirm that GRASP is the worst performing method, while the Genetic Algorithm is clearly the best method.

# 5 Conclusion and future research directions

In this paper, we consider a two-level supply-chain problem where a manufacturer and a 3PL provider cooperate to satisfy the customer demands. The manufacturer faces inventory costs, vehicle costs and penalty costs, paid to the customer for late delivery. The 3PL provider faces routing cost and penalty cost, paid to the manufacturer for extra-late delivery. In this paper, we consider a scenario where the manufacturer dominates the 3PL provider, by imposing the number of vehicles used, the batch composition and the departure times of the vehicles.

A Mixed Integer Linear Programming model is proposed for the problem of the manufacturer and the 3PL provider. This MILP can only solve very small instances. Metaheuristics are proposed to solve the manufacturer problem: a GRASP algorithm, a Tabu Search and a Genetic Algorithm. Computational results are provided. The results show that the Genetic Algorithm outperforms the other methods.

For future research, other cooperation scenarios could be investigate, such as the scenario where the 3PL provider dominates, thus imposing the departure dates for a fixed number of vehicles. Another interesting case is the scenario where the two agents cooperate in order to see if a global profit can be realized.

# References


[Cakici et al., 2014] Cakici, E., Mason, S. J., Geismar, H. N., and Fowler, J. W. (2014). Scheduling parallel machines with single vehicle delivery. *Journal of Heuristics*, 20(5):511–537.

[Chen, 2010] Chen, Z.-l. (2010). Integrated Production and Outbound Distribution Scheduling: Review and Extensions. *Operations Research*, 58(1):130–148.

[Condotta et al., 2013] Condotta, A., Knust, S., Meier, D., and Shakhlevich, N. V. (2013). Tabu search and lower bounds for a combined production-transportation problem. *Computers and Operations Research*, 40(3):886–900.

[Fu, 2014] Fu, L. (2014). *Coordination of production and distribution scheduling.* PhD thesis, Université Paris-Dauphine, Paris.

[Geismar et al., 2008] Geismar, J. H., Laporte, G., Lei, L., and Sriskandarajah, C. (2008). The integrated production and transportation scheduling problem for a product with a short lifespan. *INFORMS Journal on Computing*, 20(1):21–33.

[Glover, 1989] Glover, F. (1989). Tabu Search - Part I. *ORSA Journal on Computing*, 1(3):4–32.

[Kergosien et al., 2017] Kergosien, Y., Gendreau, M., and Billaut, J.-C. (2017). Benders decomposition based heuristic for a production and outbound distribution scheduling problem with strict delivery constraints. *European Journal of Operational Research*, 262(1):287–298.

[Koç and Sabuncuoglu, 2017] Koç, U. and Sabuncuoglu, I. (2017). Coordination of inbound and outbound transportation schedules with the production schedule. *Computers & Industrial Engineering*, 103:178–192.

[Moons et al., 2017] Moons, S., Ramaekers, K., Caris, A., and Arda, Y. (2017). Integrating production scheduling and vehicle routing decisions at the operational decision level: A review and discussion. *Computers & Industrial Engineering*, 104:224–245.

[Prins, 2004] Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers and Operations Research*, 31(12):1985–2002.

[Rohmer et al., 2015] Rohmer, S., Brain, A., Morin, P.-A., and Billaut, J.-C. (2015). A two-agent model for production and outbound distribution scheduling. In *27th European Conference on Operational Research (EURO 2015)*, Glagow.

[Ullrich, 2013] Ullrich, C. A. (2013). Integrated machine scheduling and vehicle routing with time windows. *European Journal of Operational Research*, 227(1):152–165.

[Viergutz and Knust, 2014] Viergutz, C. and Knust, S. (2014). Integrated production and distribution scheduling with lifespan constraints. *Annals of Operations Research*, 213(1):293–318.