

A Benders decomposition-based heuristic for a production and outbound distribution scheduling problem with strict delivery constraints

Yannick Kergosien, Michel Gendreau, Jean-Charles Billaut

▶ **To cite this version:**

Yannick Kergosien, Michel Gendreau, Jean-Charles Billaut. A Benders decomposition-based heuristic for a production and outbound distribution scheduling problem with strict delivery constraints. European Journal of Operational Research, 2017, 262 (1), pp.287-298. 10.1016/j.ejor.2017.03.028 . hal-03018088

HAL Id: hal-03018088
https://hal.science/hal-03018088

Submitted on 22 Feb 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Benders decomposition-based heuristic for a production and outbound distribution scheduling problem with strict delivery constraints

Y. Kergosien[1*], M. Gendreau[2], J.-C. Billaut[1]

1 : Université François-Rabelais de Tours, CNRS, LI EA 6300, OC ERL CNRS 6305,
64 avenue Jean Portalis, 37200 Tours, France
* : yannick.kergosien@univ-tours.fr / +33 2 47 36 14 23
2 : Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport (CIRRELT)
and Département de mathématiques et de génie industriel, École Polytechnique de Montréal,
C.P. 6079, succursale Centre-ville, Montréal (Québec), Canada, H3C 3A7.

February 22, 2022

Abstract - The problem addressed in this paper is from a chemotherapy production and delivery environment, where production and delivery are strongly connected problems. Independent jobs have to be prepared by pharmacy technicians working in parallel. These jobs represent pouches of injectable chemotherapy preparations. The production process corresponds to a classic parallel machine scheduling problem. Then, the jobs must be delivered to the patients by a given due date. Only one person ensures all the deliveries, making several trips between the pharmacy production unit and the patient locations. We model this step as a multi-trip traveling salesman problem, where only one salesman can make more than one trip. The objective to minimize is the maximum tardiness of delivery. In addition to the constraints that link the two problems, some constraints related to the chemical stability of chemotherapy drugs have to be taken into account: The time between the production starting time and the date the treatment is administered to the patient (here, the delivery time) cannot exceed the stability duration, as the drug may otherwise become dangerous or ineffective for the patient. Due to these constraints, the problem is more difficult to solve. The proposed resolution method in this paper is a Benders decomposition-based heuristic that makes it possible to find feasible solutions and lower bounds. The advantage of the Benders decomposition approach is that this method exploits the structure of the problem, which can be easily decomposed into two stages. Computational experiments are conducted, and a comparison with a direct exact resolution shows the efficiency of this approach.

Keywords: scheduling ; Benders decomposition ; transportation ; chemotherapy.

1 Introduction

Research in the field of Supply Chain Management has received a great deal of interest in the last two decades. An important segment of the supply chain literature deals with the integration of production and distribution problems; the reduction in costs can be significant for companies if these problems are considered simultaneously. Most of the papers in this literature consider these

integrated problems at a tactical or strategic level. At the strategic level, the production problem consists in defining the location of factories and production quantities per period of time. The distribution problem consists in designing the distribution network, i.e., the location of warehouses and retailers, and assigning production to sites. Most of the time, inventory costs play an important role and have an impact on the cost functions that must be minimized. The reader interested in these models can read the survey paper by Sarmiento and Nagi [Sarmiento and Nagi, 1999], the survey paper by Erenguc, Simpson, and Vakharia [Erenguc at al., 1999] and the review by Vidal and Goetschalckx [Vidal and Goetschalckx, 1997] or refer to [Fahimnia et al., 2012] for a specific resolution approach. A general framework for solving a coordinated production and transport scheduling problem in the supply chain is proposed by Bonfill, Espuna and Puigjaner [Bonfill et al., 2008]. [Amaro and Barbosa-Póvoa., 2008] propose a mixed-integer linear programming formulation for the optimal scheduling of industrial supply chains, taking into account the supply chains topology, the processes, the operability requirements and the different transportation policies and market conditions. To reduce storage costs and delivery lead times, a cross-docking strategy can be implemented [Dondo et al., 2011]. It consists in receiving the products at the inbound docks, quickly sorting the products based on customer demand and loading the products directly into outbound trucks for delivery. A recent survey is available in [Van Belle et al., 2012]. According to [Dondo and Cerda., 2014], some products are better suited to cross-docking like, refrigerated pharmaceuticals products.

We consider in this paper an integrated production and distribution scheduling problem at the operational level, i.e., production scheduling and vehicle routing integrated problems. The products that enter into the production process have a limited lifespan and have to be delivered before a given deadline, after their production is started. Furthermore, a delivery due date is associated to each product. This study is a continuation and extension of previous works dealing with the production planning and tracking of chemotherapy drugs for cancer treatment in the bio-pharmaceutical unit of the Oncology Clinic of the Hospital Center of Tours (France) (see [Kergosien and al., 2011] and [Mazier and al., 2010]). The entire production process of chemotherapy preparations is described in [Kergosien and al., 2011]. Software for production planning support has been developed and is used in the production center. The software treats only the scheduling problem without considering the distribution phase.

We model the production of the preparations as a classic *uniform parallel machines* scheduling problem. In our case, a machine corresponds to a pharmacy technician and a job to a chemotherapy preparation (a pouch for injection). A set of parallel machines have to perform a set of jobs and the machines are uniform, i.e., each machine has its own processing speed. A machine can process only one job at a time. Each job is characterized by a processing time and must be processed on a single machine. Each job has to be delivered to the patient at a desired time for the injection fixed by the doctor called the delivery due date. For the distribution part, only one delivery man is available to deliver the preparations to the patients. The routing problem can be defined as a multi-trip traveling salesman problem with a single salesman, or more generally as a multi-trip vehicle routing problem with a single vehicle of unlimited capacity. The objective of the problem is to minimize the maximum tardiness of delivery, which is related to the delivery due date. Another important element is the stability constraint of the chemotherapy drugs. For such drugs, the duration between the production starting time and the time of delivery for injection cannot exceed the stability time; otherwise, the drug may become dangerous or ineffective. In such a case, the drug is destroyed. This type of constraint makes the general problem harder to solve.

Because of the problem structure, a decomposition into two sub-problems seems to be an interesting approach. Among these kinds of approaches, Benders decompositions [Benders, 1962] have already demonstrated their efficiency in solving such difficult problems in the literature.

There is a vast amount of literature dealing with individual problems of production scheduling and vehicle routing, but the integrated approach to both problems at an operational level (integrated scheduling and vehicle routing) has only been addressed to a large extent in the recent literature. Most papers consider simple delivery considerations, i.e. no consideration of a vehicle routing problem and only direct deliveries or vehicles with capacities equal to one or two jobs or with only one or two destinations. In [Chen, 2004], the author proposes an integration of production and distribution operations at the tactical and operational levels. Several models are identified depending on the level decisions and the structure of production-distribution integration. According to this paper, our problem is in the category of the "*operational EPD (explicit production-distribution) model*" plus "*integration of production and outbound transportation*" and "*finite horizon with multiple time periods*", identified as class 5. This survey is extended in [Chen, 2010], where Z-L. Chen proposes a unified notation for such integrated problems, as well as an overview of different resolution approaches. In [Geismar at al., 2008], the authors consider a plant with a constant production rate for a single product with limited lifespan (constant maximum delay between delivery and production completion). Customers have different location sites and different demands to be satisfied. The objective is to minimize the makespan for satisfying all the demands. One vehicle with limited capacity is used. The authors propose lower bounds for the makespan. Then, they consider a particular case where the permutation of customers is given and propose a polynomial time algorithm. A memetic algorithm is proposed for the general problem. In [Lee and Chen, 2001], the authors study a machine scheduling problem with two types of transportation consideration. The first one considers the transportation times of jobs from one machine to another. In the second one, jobs have to be processed on a set of machines (parallel or series machines), and then the jobs must be delivered to customers. However, the transportation time to customers is assumed to be the same for each job. For these two types of models, the authors propose complexity analysis and polynomial or pseudo-polynomial resolution algorithms. In [Chen et al., 2009], H-K. Chen, C-F. Hsueh and M-S. Chang propose a nonlinear mathematical model for an integrated scheduling and vehicle routing problem with time windows and perishable food products. The demands are stochastic, the revenue of the supplier is uncertain and the objective is to maximize its expected total profit. Despite some common characteristics, this problem is quite unlike ours. In [Ullrich, 2013], C.A. Ullrich considers an integrated production and outbound distribution problem with the objective of minimizing total lateness. The scheduling problem is a parallel machine scheduling problem with machine-dependent ready times, and a fleet of heterogeneous vehicles are considered for the vehicle routing problem. Jobs have to be delivered in time windows. The author proposes an MILP formulation, decomposition heuristics and a genetic algorithm. The first decomposition approach first solves a vehicle routing problem, then a parallel machine scheduling problem, and finally another vehicle routing problem. The second decomposition approach first solves a parallel machine scheduling problem and then a vehicle routing problem. This problem is somewhat similar to the problem that we consider except that we consider maximum lateness as an objective function and we have to consider the perishability of the jobs. This aspect is the main topic of [Amorim, 2013], where perishability aspects in supply chain management are analyzed comprehensively. A review of the work published on modeling perishability for production and distribution planning is performed, with one part dedicated to integrated approaches. In this part, few papers are identified, and [Chen et al., 2009] is the most relevant to our study.

Some recent papers show similarities to our study. In [Cakici et al., 2015], the authors consider a parallel machine scheduling problem in which jobs are scheduled on the machines, assigned to batches (only jobs destined for the same customer can be batched together for delivery) and de-

livered by a single vehicle with limited capacity. The objective is to minimize the total weighted delivery time. The problem is decomposed into sub-problems: machine scheduling, batch assignment and batch sequencing. An MILP formulation is proposed as well as heuristic algorithms. The problem shows similarities to our problem except for the definition of batches and the objective function, and in addition to the fact that there is no perishability constraint. In [Viergutz and Knust, 2014], C. Viergutz and S. Knust consider a single production facility modeled by a single machine. After completion, jobs have to be delivered before a given deadline (perishability) and within a time window to customers by a single vehicle. Because of limited resources, not all 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. In [Lee et al., 2014], the authors consider the case of production and delivery of medical treatments requiring radioactive substances with a half-life of minutes. The scheduling problem is treated as a special type of bin packing problem (machines are cyclotrons with different capacities). A fleet of heterogeneous vehicles is available for delivery, but split deliveries are forbidden. The authors propose a MILP formulation and heuristic algorithms. In [Belo-Filho et al., 2015], the authors consider a facility with parallel production lines, where perishable jobs are produced and have to be delivered to distinct customers in given time windows. Several vehicles are available; they can deliver jobs to different customers in a single trip. Sequence-dependent setup times and costs are considered, and the objective is to minimize production and transportation costs. The authors propose an MILP formulation of the problem, two heuristic algorithms and an adaptive large neighborhood search framework.

This paper is organized as follows. In Section 2, the problem extracted from the chemotherapy production context is described in detail, notations are introduced and an integer linear programming formulation of the problem is given. A table of notations is given in the appendix. The proposed Benders decomposition-based heuristic is presented in Section 3. Computational experiments are reported in Section 4. Finally, we conclude the study and propose directions for further research.

2 Problem description and notations

The process for receiving a chemotherapy treatment by injection – as it is done in France – is the following. After having examined a patient (to evaluate the appropriateness of treatment), the oncologist sends an electronic prescription to the bio-pharmaceutical unit where the chemotherapy preparation is produced. The prescription information specifies the type of medical chemotherapy preparation that will be administered to the patient, the exact dosage and the time schedule of administration. A production line is composed of a sterilizer and an isolator, where a pharmacy technician realizes the chemotherapy preparations for injection. Once the product is processed, a control is performed (a sample of the preparation is tested). Depending on the date of administration prescribed by the doctor, the drug must be stored in a refrigerator at a recommended temperature or be immediately delivered.

We can model the whole problem as follows, with only one assumption to simplify the problem (regarding the control process, described later). A set of n jobs have to be scheduled without interruption on m uniform parallel machines. Each job J_j, $j \in \{1, ..., n\}$ is characterized by:

- a release date r_j the execution of job J_j cannot start before this date that represents the date of treatment validation by the doctor, after an examination of the patient.

- a processing time p_j that depends on the type of chemotherapy preparation to realize. This duration corresponds to an assignment to a machine with a production rate equal to 1.
- a stability time st_j if the administration of J_j is subject to a stability constraint. After the starting time of its production, the job has to be delivered before st_j time units.
- a due date or delivery due date d_j is the desired date of the administering the treatment corresponding to job J_j.

The problem consists in assigning all the jobs to m uniform parallel machines and defining their starting time (or their sequence). Each machine M_i can process at most one job at a time without preemption, and a job is processed by only one machine. Since the pharmacy technicians do not have the same experiences and skills, each machine M_i has a production rate (yield) denoted by s_i, and $p_j \times s_i$ is the time needed to process J_j on machine M_i.

Before the next step, which consists in delivering the preparations, each job has to be controlled. This operation is the same for all jobs (same duration) and can be performed by several persons. To simplify the problem, it is assumed that each job can be analyzed directly after its production, with no waiting time. Since this control time is short in comparison with processing time, it sticks to reality. Each control takes p_A time units. We denote by C_j the production completion time of job J_j. Therefore, a job J_j that is completed at time C_j is ready for delivery at time $C_j + p_A$.

The finished jobs have to be gathered into batches.

For the multi-trip traveling salesman problem, there is only one delivery man, who is supposed to be available at all times. Each trip or route starts and ends at the same depot, which is the pharmacy production unit, and each preparation has to be delivered. The route of any given batch can only start when all the jobs of the batch have been completed and controlled. The capacity of the delivery man is considered unlimited. We denote by tt_{j_1,j_2}, $\forall j_1, j_2 \in \{0, ..., n\}$, the travel time between site j_1 and site j_2, where site $j \in \{1, ..., n\}$ corresponds to job J_j and site 0 corresponds to the pharmacy production unit.

A solution to the global problem is the assignment of jobs to the machines, the sequence of jobs on each machine, the composition of batches of delivery, and for each batch, the sequence of delivery. We denote by D_j the delivery completion time of J_j. The objective function of the problem is to minimize the maximum tardiness which is given by:

$$T_{max} = \max_{j \in \{1,...,n\}} (0, D_j - d_j)$$

Integer Linear Programming formulation

An integer linear programming model for the problem, noted ILP, is the following. We consider one list $\mathcal{L}$ of jobs for delivery, and routes are defined from this list. We define the following binary variables.

$$y^i_{j_1,j_2} = \begin{cases} 1 & \text{if job } J_{j_1} \text{ is scheduled before } J_{j_2} \text{ on machine } M_i \\ 0 & \text{otherwise} \end{cases}$$
$$\forall j_1, j_2 \in \{0, ..., n+1\}, j_1 \neq j_2, \forall i \in \{1, ..., m\}$$

Note that jobs J_0 and J_{n+1} are considered dummy jobs that are scheduled in the first and last position, respectively, on each machine.

$$x_{j,k} = \begin{cases} 1 & \text{if job } J_j \text{ is in position } k \text{ in list } \mathcal{L} \\ 0 & \text{otherwise} \end{cases}$$
$$\forall j \in \{1, ..., n\}, \forall k \in \{1, ..., n\}$$

$$N_k = \begin{cases} 1 & \text{if a new tour starts in } \mathcal{L} \text{ with the job at position } k \\ 0 & \text{otherwise} \end{cases}$$
$$\forall k \in \{1, ..., n\}$$

We define the following continuous variables.

- S_j starting time of processing job J_j, $\forall j \in \{1, ..., n\}$,
- D_j delivery completion time of J_j, $\forall j \in \{1, ..., n\}$,
- t_k starting time of the tour delivering the job in position k, $\forall k \in \{1, ..., n\}$
- $T_{\max}$ the maximum tardiness of delivery.

The objective function is to minimize maximum tardiness.

$$\text{Minimize } T_{\max} \tag{1}$$

The following constraints concern the scheduling problem.

On machine M_i, the first and last dummy jobs have only one predecessor and successor ($\forall i \in \{1, ..., m\}$):

$$\sum_{j=1}^{n+1} y^i_{0,j} = 1 \tag{2}$$

$$\sum_{j=0}^{n} y^i_{j,n+1} = 1 \tag{3}$$

A job cannot start processing before its release date ($\forall j \in \{1, ..., n\}$):

$$S_j \geq r_j \tag{4}$$

If job J_{j_1} is scheduled before J_{j_2} on machine M_i ($\forall j_1, j_2 \in \{1, ..., n\}, j_1 \neq j_2, \forall i \in \{1, ..., m\}$), we have the following constraints. Note that $p_{j_1} s_i$ is the duration of J_{j_1} on machine M_i and M is a high value.

$$S_{j_2} \geq S_{j_1} + p_{j_1} s_i - M(1 - y^i_{j_1,j_2}) \tag{5}$$

All jobs J_j must be scheduled ($\forall j \in \{1, ..., n\}$):

$$\sum_{i=1}^{m} \sum_{\substack{j'=0 \\ j' \neq j}}^{n} y^i_{j',j} = 1 \tag{6}$$

If job J_j is assigned to machine M_i, this job has one predecessor and one successor ($\forall i \in \{1, ..., m\}$, $\forall j \in \{1, ..., n\}$):

$$\sum_{\substack{j'=0 \\ j' \neq j}}^{n} y^i_{j',j} = \sum_{\substack{j'=1 \\ j' \neq j}}^{n+1} y^i_{j,j'} \tag{7}$$

The following constraints concern the relation between the scheduling and routing problems.

If job J_j is in position k in list $\mathcal{L}$, the tour cannot start before the completion time of J_j ($\forall j \in \{1, ..., n\}$, $\forall k \in \{1, ..., n\}$):

$$t_k \geq S_j + p_A + p_j \sum_{i=1}^{m} s_i \sum_{\substack{j'=0 \\ j' \neq j}}^{n} y^i_{j',j} - M(1 - x_{j,k}) \tag{8}$$

The following constraints concern the routing problem.

A job J_j is in one position in list $\mathcal{L}$, and there is only one job per position in $\mathcal{L}$:

$$\sum_{k=1}^{n} x_{j,k} = 1 \qquad \forall j \in \{1, ..., n\} \tag{9}$$

$$\sum_{j=1}^{n} x_{j,k} = 1 \qquad \forall k \in \{1, ..., n\} \tag{10}$$

If job J_j is in position k in $\mathcal{L}$, J_j cannot be delivered before the starting time of the tour plus the transportation time from site 0 to site j ($\forall j \in \{1, ..., n\}$, $\forall k \in \{1, ..., n\}$):

$$D_j \geq t_k + tt_{0,j} - M(1 - x_{j,k}) \tag{11}$$

If J_{j_1} and J_{j_2} are consecutive in list $\mathcal{L}$, with J_{j_1} in position k and J_{j_2} in position $k+1$ ($\forall j_1, j_2 \in \{1, ..., n\}$, $j_1 \neq j_2$, $\forall k \in \{1, ..., n-1\}$), then:

$$D_{j_2} \geq D_{j_1} + tt_{j_1,j_2} - M(2 - x_{j_1,k} - x_{j_2,k+1}) \tag{12}$$

The route of the job in position $k+1$ cannot start before the route of the job in position k ($\forall k \in \{1, ..., n-1\}$):

$$t_k \leq t_{k+1} \tag{13}$$

If the position $k+1$ in $\mathcal{L}$ does not correspond to a new tour, the start time of the job in position $k+1$ is the same as for the job in position k ($\forall k \in \{1, ..., n-1\}$):

$$t_k \geq t_{k+1} - MN_{k+1} \tag{14}$$

If a new route starts with the job in position $k+1$ in $\mathcal{L}$, the start time of this route is greater than the delivery time of the jobs of tour k plus the transportation time to site 0 ($\forall j \in \{1, ..., n\}$, $\forall k \in \{1, ..., n-1\}$):

$$t_{k+1} \geq D_j + tt_{j,0} - M(2 - N_{k+1} - x_{j,k}) \tag{15}$$

Each job has to be delivered before its stability time expires ($\forall j \in \{1, ..., n\}$):

$$D_j \leq S_j + st_j \tag{16}$$

The objective function is ($\forall j \in \{1, ..., n\}$):

$$T_{\max} \geq D_j - d_j \tag{17}$$

This ILP model contains $m(n+2)^2+n^2+n$ binary variables and $3n+1$ continuous variables and $m(n^2+n+2)+n^3+2n^2+7n-2$ constraints. Some cuts can be added to facilitate the resolution of the model.

Several ILP formulations of the problem have been tested, by changing several types of variables (i.e., using position-based variables or sequence-based variables for the routing part or the scheduling part). Preliminary computational experiments on 90 small instances showed that this model is the most efficient.

To apply a classical Benders decomposition, the problem requires a certain substructure, in particular when some variables are fixed; the resulting sub-problem has to be a linear programming problem. Since the most important decision variables are binary, this model cannot be used as it is for the classical Benders decomposition method.

3 Solution approach

Benders decomposition is based on a partition of the variables into two sets X and Y. A *master problem* considers only the set of variables Y, and a sub-problem tests if the solution of the master problem also satisfies the remaining constraints by finding values for variables X. If such a solution cannot be found, a "Benders cut" is added to the master problem, and the process is iterated. Once a feasible solution of the sub-problem is found, this solution may not lead to the optimal solution for the original problem. In this case, an optimality Benders cut is added to the master problem, and the process is iterated; otherwise, the original problem is solved.

The solution approach that we propose is a hybridization of a classical Benders decomposition algorithm and a heuristic dedicated to this combined routing and scheduling problem. Before introducing the general algorithm, we present the classical Benders decomposition scheme and then the Benders decomposition that is applied to our problem. In the end, the general hybrid algorithm is described, as well as several improvement techniques.

3.1 Benders decomposition

First, we succinctly present the classical Benders decomposition algorithm, and then its application to our problem. Since some assumptions have to be made, the resulting algorithm presented in this section gives only a lower bound of the problem, from which we can deduce an upper bound.

3.1.1 Classical algorithm

Let P be an integer programming problem modeled as follows. x is a vector of continuous variables and y is a vector of integer variables, which can take a value in the set $\mathcal{D}_y$. f is a linear function, g is a mapping, c and a are vectors, and A is a matrix.

$$\begin{aligned} \text{Problem } P : \text{Minimize } & f(y) + cx \\ \text{Subject to} : \ & g(y) + Ax \geq a \\ & y \in \mathcal{D}_y, \ x \geq 0 \end{aligned}$$

If we consider a feasible assignment y^* for the y variables, the following problem has to be solved to find the values of the x variables (we call this problem *the slave problem SP*):

$$\begin{aligned} \text{Problem } SP : \text{Minimize } & cx \\ \text{Subject to : } & Ax \geq a - g(y^*) \\ & x \geq 0 \end{aligned}$$

Consider the dual problem of the slave problem, denoted DSP (u are the dual variables):

$$\begin{aligned} \text{Problem } DSP : \text{Maximize } & u(a - g(y^*)) \\ \text{Subject to : } & uA \leq c \\ & u \geq 0 \end{aligned}$$

The main concept of the Benders decomposition is to find an assignment of variables y in a master problem MP and then check if a feasible solution exists for the slave problem and if this solution with the assignment of y leads to the optimal solution for the original problem P. If no feasible solution is found, an infeasibility Benders cut is added to the master problem. Otherwise, an optimality Benders cut is added to the master problem. Both types of Benders cuts are intended to avoid some assignments of variables y. The master problem MP is defined as follows:

$$\begin{aligned} \text{Problem } MP : \text{Minimize } & z \\ \text{Subject to : } & z \geq f(y) + u_i^*(a - g(y)) \ \forall i \in I \\ & u_j^*(a - g(y)) \leq 0 \ \forall j \in J \\ & u \geq 0 \end{aligned}$$

with I the set of iterations where the solution of SP does not lead to the optimal solution for P, J the set of iterations where SP is not feasible, and u_i^* the values of the variables of DSP. The general algorithm is the following.

To apply this method to our problem, we consider the production scheduling part as the *slave problem* and the delivery part as the *master problem*. However, the issue is that the slave problem should be a linear program (solvable in polynomial time to optimality), but the scheduling problem considered on uniform parallel machines is NP-hard and cannot be modeled as a linear program. To overcome this difficulty, we decided to relax the constraint of non-preemption in order to make the problem solvable in polynomial time. Therefore, the obtained method makes it possible to find a lower bound of the problem. This lower bound is used to find a feasible solution, thanks to an additional heuristic (see Section 3.2). There are two main reasons for choosing the production part of the problem as the slave problem. First, the quality of the relaxation of the production part is better than the quality of the relaxation of the distribution part. Thus, the Benders cuts are more efficient and the lower bound is better. Second, the Benders decomposition is better when the master problem has relatively few integer variables. In addition, the number of binary variables is larger in the production scheduling part than in the distribution part.

Thus, the ILP model is decomposed into two models. The first model that represents the slave problem finds a solution to the scheduling problem. It corresponds to variables $y^i_{j_1,j_2}$ and to constraints (2) to (7). The second model that represents the master problem solves the routing problem. It corresponds to variables $x_{j,k}$ and N_k and constraints (9) to (17).

Algorithm 1 General Benders algorithm

$End \leftarrow false$
while $End \neq true$ **do**
  Solve the master problem MP
  if MP is infeasible **then**
    return *"problem P is infeasible"*; $End \leftarrow true$
  else if MP is unbounded **then**
    return *"problem P is unbounded"*; $End \leftarrow true$
  else
    let (y^*, z^*) the optimal solution
  end if
  Solve the dual slave problem DSP
  if DSP is infeasible **then**
    return *"problem P is infeasible or unbounded"*
  else if DSP is feasible with u^* as optimal solution such as $z^* < f(y^*) + u^*(a - g(y^*))$ **then**
    add $z \geq f(y) + u^*(a - g(y))$ to MP
  else if DSP is feasible with u^* as optimal solution such as $z^* \geq f(y^*) + u^*(a - g(y^*))$ **then**
    return (x^*, y^*) *as the optimal solution of P*; $End \leftarrow true$
  else if DSP is unbounded **then**
    there is an extreme direction u^* such that $u^*(a - g(y^*)) > 0$, add the cut $u^*(a - g(y)) \leq 0$ to MP
  end if
end while

3.1.2 The slave problem

The slave problem is a uniform parallel machine scheduling problem with preemption. At this step, a solution of the master problem is given, and thus the following elements are known:

- the order in which the jobs are delivered σ_k denotes the index of the job in position k in list $\mathcal{L}$.
- the date pd_k at which the job J_{σ_k} in position k is ready for control and delivery; these dates can be considered pseudo due dates for the scheduling problem.
- the delivery time D_k of the job in position k; from this date, we deduce a new release date for the job delivered in position k in list $\mathcal{L}$, noted $R_k = \max(r_{\sigma_k}, D_{\sigma_k} - st_{\sigma_k})$. This release date takes the stability time of J_{σ_k} into account.
- P_k the processing time of the job delivered in position k in list $\mathcal{L}$, i.e. p_{σ_k}.

In this problem, each machine has its production rate, and we assume that preemption is allowed. To each job is associated a release date, a processing time and a due date. The linear programming model of the slave problem is based on the model of [Lawler and Labetoulle, 1978]. In this model, the tasks are indexed in their increasing due date order, which is implicitly the case in our model because the jobs are indexed in their delivery position increasing order. Some elements are added to the original model in order to integrate the production rate of the machines and the release dates. Note that the method presented in [Brucker, 2007] based on a max flow computation cannot be used in our slave problem; in this method, it is necessary to perform a pre-processing step on the data, which cannot be executed during the Benders decomposition process. This step

consists in ordering and mixing all different "release dates - values" (R_k) and "due dates - values" (pd_k) in order to obtain several consecutive time intervals that will be used as nodes in the flow network. The decision variables correspond to the quantity of jobs processed on each time interval. However, the Benders cuts that are used in the master problem are based on the dual variables of the flow, which are associated to unknown time intervals for the master problem.

Consider $pt^l_{i,k} \geq 0$, $\forall l \in \{1, ..., n\}$, $\forall i \in \{1, ..., m\}$, $\forall k \in \{1, ..., n\}$, the continuous variables that represent the total amount of time that machine M_i works on the job at position k during time period $[pd_{l-1}, pd_l]$ (note that $l = 0$ corresponds to a dummy job for which $pd_0 = P_0 = R_0 = 0$). The slave problem consists in finding a feasible solution that respects the following constraints.

$$\text{Slave problem} : \text{Find } pt^l_{i,k}$$

$$\text{Subject to} : \sum_{i=1}^{m}\sum_{l=1}^{k}\frac{pt^l_{i,k}}{s_i} \geq P_k \qquad \forall k \in \{1, ..., n\} \tag{18}$$

$$\frac{1}{\max_{j\in\{1..n\}} p_j}\sum_{i=1}^{m}\frac{pt^l_{i,k}}{s_i} \leq A^l_k \qquad \forall l \in \{1, ..., n\}\ \forall k \in \{l+1, ..., n\} \tag{19}$$

$$\sum_{i=1}^{m} pt^l_{i,k} \leq pd_l - pd_{l-1} \qquad \forall l \in \{1, ..., n\}\ \forall k \in \{l, ..., n\} \tag{20}$$

$$\sum_{k=l}^{n} pt^l_{i,k} \leq pd_l - pd_{l-1} \qquad \forall l \in \{1, ..., n\}\ \forall i \in \{1, ...m\} \tag{21}$$

The constraints (18) state that at least the total amount of time P_k for the job in position k is processed. The aim of constraints (19) is to take into account the release date of a job. A^l_k is equal to 1 if $R_k < pd_l$ and 0 otherwise. If the release date of a task σ_k is greater than pd_l, no quantity of job J_{σ_k} can be scheduled on any machine before pd_l. The weakness of this model is when the release date R_k belongs to a period $[pd_{l-1}, pd_l]$. With these constraints, the period $[pd_{l-1}, R_k]$ may contain some quantity of job J_{σ_k}, although it should be impossible. Unfortunately, as far as we know, it is not possible with this model to take the release dates into account in a precise way. In any case, since we are looking for a lower bound, this assumption is not an issue. Constraints (20) guarantee that for each period and each job, the quantity on the machines should not exceed the width of the period. Constraints (21) state that for each period and each machine, the total duration of each job cannot exceed the length of the period.

The dual of the slave problem is represented by the following objective function:

$$\text{Maximize } \sum_{k=1}^{n} u_k(-P_k) + \sum_{l=1}^{n}\sum_{k=l+1}^{n} s^l_k(A^l_k) + \sum_{l=1}^{n}\sum_{k=l}^{n} v^l_k(pd_l - pd_{l-1}) + \sum_{l=1}^{n}\sum_{i=1}^{m} w^l_i(pd_l - pd_{l-1}) \tag{22}$$

with u_k, s^l_k, v^l_p and w^l_i the dual variables.

Since this is only a feasibility problem, the Benders cuts to add to the master problem, when the dual of the slave problem is unbounded are as follows:

$$\sum_{k=1}^{n} \overline{u_k}(-P_k) + \sum_{l=1}^{n}\sum_{k=l+1}^{n} \overline{s^l_k}(A^l_k) + \sum_{l=1}^{n}(pd_l - pd_{l-1})(\sum_{k=l}^{n}\overline{v^l_k} + \sum_{i=1}^{m}\overline{w^l_i}) \leq 0 \tag{23}$$

with $\overline{u_k}$, $\overline{s_k^l}$, $\overline{v_k^l}$ and $\overline{w_i^l}$ the values of the dual variables that represent the extreme direction (note that P_k, R_k, pd_k and A_k^l are variables for the master problem).

3.1.3 The master problem

As presented in Section 2, the integer linear programming model of the master problem is also based on a sequence of jobs, noted $\mathcal{L}$, sorted by their delivery position. The master problem consists in finding this sequence, the number of trips, when they appear in this sequence and the departure time of each trip. The definitions of variables $x_{j,k}$, N_k, D_j, t_k and $T_{\max}$ in the previous model are unchanged. The master problem is also linked to the slave problem and the Benders cuts, thanks to the following variables:

- pd_k, $\forall k \in \{1, ..., n\}$: expected finishing time of the job at position k (ready for control and delivery).
- P_k, $\forall k \in \{1, ..., n\}$: processing time of the job in position k.
- R_k, $\forall k \in \{1, ..., n\}$: release date at which the job in position k is available for processing on machines, considering the release time and the stability time.
- $A_k^{k'}$, $\forall k, k' \in \{1, ..., n\}$: $\begin{cases} 1 & \text{if } R_k < pd_{k'}, \\ 0 & \text{otherwise } (R_k \geq pd_{k'}). \end{cases}$

The model of the master problem, without the Benders cuts as defined in (23), is the following:

$$\text{Master problem} : \text{Minimize } T_{\max}$$

$$\text{Subject to} : (9) \text{ to } (15), (17) \tag{24}$$

$$t_k \geq \sum_{j=1}^{n} r'_j x_{j,k} \qquad \forall k \in \{1, ..., n\} \tag{25}$$

$$LB_{T_{max}} \leq T_{max} \leq UB_{T_{max}} \tag{26}$$

$$R_k + P_k \leq pd_k \qquad \forall k \in \{1, ..., n\} \tag{27}$$

$$pd_k = t_k - p_A \qquad \forall k \in \{1, ..., n\} \tag{28}$$

$$P_k = \sum_{j=1}^{n} p_j x_{j,k} \qquad \forall k \in \{1, ..., n\} \tag{29}$$

$$R_k \geq \sum_{j=1}^{n} r_j x_{j,k} \qquad \forall k \in \{1, ..., n\} \tag{30}$$

$$R_k \geq d_j - st_j - M(1 - x_{j,k}) \qquad \forall k, \forall j \in \{1, ..., n\} \tag{31}$$

$$A_k^{k'} \geq \frac{pd_{k'} - R_k}{M} \qquad \forall k,\ k' \in \{1, ..., n\}\ , k \neq k' \tag{32}$$

$$A_k^{k'} < 1 + \frac{pd_{k'} - R_k}{M} \qquad \forall k,\ k' \in \{1, ..., n\}\ , k \neq k' \tag{33}$$

The first part of the model uses the same constraints as for the delivery part in the ILP model ((9) to (15) and (17)). Even if the starting times of trips are strongly connected to the slave problem and to the Benders cuts, constraints (25) guarantee that the starting time of a trip is at least equal to the maximum of the earliest ready times of the jobs in that trip. These earliest ready times for delivery, noted r'_j, depend on the release dates, processing times on the fastest machine and the

time to analyze: $r'_j = r_j + \min_{i\in\{1,...,m\}} s_i p_j + p_A$. Constraints (26) bound the $T_{\max}$ with a lower bound $LB_{T_{\max}}$ and an upper bound $UB_{T_{\max}}$. Constraints (27) ensure that there is enough time to process the job at position k between R_k and pd_k. Constraints (28) define the variables pd_k with the starting times of trips, and constraints (29) define the variables P_k according to the position of each job. Constraints (30) and (31) determine a global release date of each job according to the initial release date and the stability time. Finally, constraints (32) and (33) define the Boolean variables $A_k^{k'}$.

Note that when a solution is found, there is no guarantee that the variables R_k are set to their smallest possible values. This may be restrictive for the slave problem. Thus, before solving the slave problem, the variables R_k are set to their smallest possible values (respecting all the constraints) and the variables $A_k^{k'}$ are updated.

The master problem resolution step is the most time consuming. Thus, some pre-processing and cut generation are used in order to accelerate the resolution. The first set of cuts are intuitive:

$$r'_j + tt_{0,j} \leq D_j \qquad \forall j \in \{1,...,n\} \qquad (34)$$

$$t_k + 2 \min_{j\in\{1,...,n\}} (tt_{0,j}) N_{k+1} \leq t_{k+1} \qquad \forall k \in \{1,...,n-1\} \qquad (35)$$

$$\sum_{j=1}^{n} r'_j x_{j,k} \leq t_{k'} \qquad \forall k \in \{1,...,n-1\}\ \forall k' \in \{k+1,...,n\} \qquad (36)$$

$$t_k + \sum_{j=1}^{n} (tt_{0,j} - d_j) x_{j,k} \leq T_{max} \qquad \forall k \in \{1,...,n\} \qquad (37)$$

Constraints (34) state that each preparation cannot be delivered before $r'_j + tt_{0,j}$ time units, assuming that there is no waiting time between the end of the analysis and the starting time of the job delivery. Constraints (35) imply a minimum travel time between the starting time of the trip delivering the preparation at position k and the trip delivering the preparation at position $k+1$, if a new trip starts at position $k+1$. Constraints (36) state that if a job J_j is delivered in position k, all starting times of the next trips, $k' > k$, are greater than r'_j. Finally, constraints (37) make it possible to deduce a lower bound of the maximum tardiness if a job J_j is delivered at position k. This lower bound is based on an estimated delivery time that is equal to the starting time of the trip delivering the job J_j plus the travel time to directly deliver J_j.

The last part of pre-processing and cut generation is based on the following reasoning. Let α be the sequence of jobs sorted in r'_j increasing order and β the sequence of jobs sorted in $(d_j - tt_{0,j})$ increasing order, and let α_k and β_k represent the index of the job at position k in α and β, respectively. Based on the α sequence, the starting time of the trip delivering the job at position k cannot be lower than the k^{th} smaller completion time of the production stage, i.e.:

$$r'_{\alpha_k} \leq t_k \qquad \forall k \in \{1,...,n\} \qquad (38)$$

From these lower bounds on the trip starting times and an upper bound of the maximum tardiness $UB_{T_{max}}$, we can set some variables $x_{j,k}$ to 0 as follows:

$$\forall l \in \{1,...,n\}\ \forall j \in \{1,...,n\} : \text{if } r'_{\alpha_l} + tt_{0,j} - d_j \geq UB_{T_{max}} \text{ then } \forall k \in \{l,...,n\} : x_{j,k} = 0 \qquad (39)$$

This pre-processing generation needs to have a good upper bound for the $T_{\max}$ (presented in the next section). The main idea is to deduce which jobs cannot be placed too late in the sequence,

i.e., those that would increase the maximum tardiness of the best solution found. The same idea can be generalized for the criterion $T_{\max}$ by adding the following cuts:

$$\forall l \in \{1,...,n\}\ \forall j \in \{1,...,n\} \text{ such that } r'_{\alpha_l} + tt_{0,j} - d_j > 0,\ \forall k \in \{l,...,n\} : x_{j,k} \leq \left\lfloor \frac{T_{max}}{r'_{\alpha_l} + tt_{0,j} - d_j} \right\rfloor \tag{40}$$

Based on the same idea, it is possible to reverse the reasoning by using list β instead of α. In this case, the goal is to deduce which preparation should not be placed too early in the sequence because it would cause a delivery of another preparation to be too late.

$$\forall l \in \{1,...,n\}\ \forall j \in \{1,...,n\} : \text{if } r'_j + tt_{0,\beta[l]} - d_{\beta[l]} \geq UB_{T_{max}} \text{ then } \forall k \in \{1,...,l\} : x_{j,k} = 0 \tag{41}$$

$$\forall l \in \{1,...,n\}\ \forall j \in \{1,...,n\} \text{ such that } r'_j + tt_{0,\beta[l]} - d_{\beta[l]} > 0,\ \forall k \in \{1,...,l\} : x_{j,k} \leq \left\lfloor \frac{T_{max}}{r'_j + tt_{0,\beta[l]} - d_{\beta[l]}} \right\rfloor \tag{42}$$

3.2 The Benders decomposition-based heuristic

This section presents the *Benders decomposition-based heuristic* that is based on the Benders decomposition algorithm with the master and slave problem previously detailed. The general algorithm is first presented, and four important steps of the algorithm are then described.

3.2.1 General algorithm

Algorithm 2 describes the main steps of the Benders decomposition-based heuristic (BDH). First, two initial procedures are executed, which compute a lower bound and some initial Benders cuts (detailed later). The main loop contains the iteration procedure for the resolution of the master and slave problem as in any classic Benders decomposition algorithm, except for lines 7 and 8 and lines 12 to 22. An upper bound is computed using a heuristic algorithm (lines 7 and 8) that returns a feasible solution (detailed later). It consists in finding a solution for the production part without constraint relaxation (i.e., preemption is not allowed) from a current solution of the master problem (the delivery part). Finally, lines 12 to 22 are executed when a solution is found for the slave problem. If this lower bound is equal to the upper bound, the algorithm returns the optimal solution. Otherwise, another iteration is executed to explore new solutions as a result of the heuristic algorithm of line 7. However, if the set of Benders cut do not change, the same upper bound of the previous iteration is found. To avoid useless iteration, when a solution is found for the slave problem (and no Benders cut is added), an active Benders cut is removed in order to reiterate between the master problem and the slave problem and to find new solutions for the master problem and for the heuristic algorithm. If no active Benders cut appears in the last resolution of the master problem, the algorithm returns the best solution found. The BDH finishes after NB_ITE_MAX deletions of an active Benders cut.

3.2.2 Lower bound computation

The lower bound computation described by Algorithm 3 is based on a dichotomic search, assuming that the preparations can be delivered directly after their analysis and the constraint of stability is relaxed. To solve the scheduling problem, the ILP model of the slave problem is used.

Algorithm 2 General algorithm

$LB \leftarrow$ Compute a lower bound
$UB \leftarrow$ High value
Add initial Benders cuts using the relaxed master problem
$i \leftarrow 0$
while $i \leq NB_ITE_MAX$ **do**
  Solve the master problem
  $UB_{heur} \leftarrow$ Find an upper bound using a heuristic algorithm
  $UB \leftarrow \min(UB, UB_{heur})$
  Solve the slave problem
  if the slave problem has not a feasible solution **then**
    Add a Benders cut
  else
    $LB \leftarrow$ the result of the previous resolution of the master problem
    if $UB = LB$ **then**
      return UB (optimal solution)
    else
      if There is at least one active Benders cut **then**
        Remove one active Benders cut selected randomly
        $i \leftarrow i + 1$
      else
        return UB
      end if
    end if
  end if
end while

Algorithm 3 Lower bound computation algorithm

$L_{LB} \leftarrow 0$
$U_{LB} \leftarrow$ High value
$LB \leftarrow (L_{LB} + U_{LB})/2$
while $L_{LB} \neq U_{LB}$ **do**
  for $k \in \{1, .., n\}$ **do**
    $pd_p \leftarrow d_k + LB - tt_{0k} - p_A$
  end for
  Solve the slave problem considering that the jobs are sorted in pd_p increasing order
  if the slave problem has a feasible solution **then**
    $U_{LB} \leftarrow LB$
    $LB \leftarrow (L_{LB} + U_{LB})/2$
  else
    $L_{LB} \leftarrow LB$
    $LB \leftarrow (L_{LB} + U_{LB})/2$
  end if
end while
return LB

3.2.3 Initial Benders cuts

A famous technique to accelerate the Benders decomposition algorithm is to generate a set of initial cuts from the linear programming relaxation of the master problem [McDaniel and Devine, 1977]. It consists in iteratively solving the linear programming relaxation of the master problem and the slave problem in the same way as the Benders decomposition algorithm until an optimal solution is found. Each iteration where the slave problem has no feasible solution gives an initial Benders cut. The advantage of this technique is that the iteration process is quite fast since two LP problems are solved, but the number of iterations can be large before finding an optimal solution. Thus, the number of iterations is limited to $2n$. This phase may also improve the lower bound.

3.2.4 Tabu Search heuristic to build a feasible solution

The heuristic builds a feasible solution for the scheduling problem from a solution of the master problem. The scheduling problem is the same as for the slave problem, but preemption is not allowed. The heuristic also considers the maximum tardiness to minimize. Batch composition and sequences of delivery are known, and a direct coding is implemented: A solution of the scheduling problem is represented by a sequence of jobs on each machine and is evaluated according to the solution of the master problem. Depending on how the jobs are scheduled on the machines, the starting time of the trips and the delivery times of the jobs can be deduced, which gives the maximum tardiness.

The scheduling problem is solved using a Tabu Search heuristic [Glover, 1986] in order to provide a sufficiently quick and efficient solution. Its structure is classic; solutions are explored in the search space one by one. The Tabu Search uses a neighborhood search procedure to iteratively move from one solution to another by choosing the best solution that is not tabu until a stopping criterion is reached. In our case, the stopping criterion is a number of iterations without improvement, denoted by Ite_{tabu}.

A list algorithm has been developed to build an initial solution. All jobs are ordered in a list in their batch delivery order (i.e., the set of jobs of the first batch delivery are scheduled first in the list and so on). For each batch, the jobs with a stability time constraint are ordered first in r_j non decreasing order, and the remaining jobs in a second part in r_j non decreasing order. Jobs are scheduled in this order in the time slot minimizing the maximum tardiness (because of the release times, the schedule may have some idle times, and several insertion positions are tested).

The neighborhood operator used in the Tabu Search is the INSERT operator that consists in removing a job from one position in the sequence on a machine and inserting it at another position on the same machine, or on another machine. To build the set of neighbors, all jobs are tested on all possible positions. Only feasible solutions are considered (solutions violating stability constraints are removed). If no feasible solution exists in the neighborhood, the process is restarted after emptying the tabu list.

The tabu list contains a set of forbidden moves and has a limited size noted as $Size_{tabu}$. A move is defined by the job number and the last assigned machine number. If the tabu list contains a move defined by a job J_j and a machine M_i, all insertions of the job J_j on machine M_i are prohibited during neighborhood construction. However, a tabu move can be accepted using aspiration criteria. The aspiration criterion consists in allowing a tabu move when the result gives a new solution with an objective value better than the current best-known solution.

4 Computational experiments

In this section, the performances of the Benders decomposition-based heuristic (BDH) are evaluated on pseudo-real instances. The algorithms are implemented in C++ language. Tests have been performed on an Intel(R) Core(TM)2 CPU T7200 @ 2.00GHz with 2G of Ram, and CPLEX 12.5 is used as an MILP solver.

4.1 Instances

To evaluate the efficiency of the method, a set of random data are created, corresponding to the real activity of the chemotherapy production unit of the Oncology Clinic of the hospital complex of Tours. This activity is very similar to many other units of chemotherapy production. The parameters of generation were also chosen in order to make instances difficult to solve. For example, if the processing times of chemotherapy preparations are large in comparison with the travel times between the delivery sites, the delivery part of the problem will be easy. It is sufficient to solve the scheduling problem, and then finding the trips of delivery is easy. Similarly, too big values for the travel times in comparison with the processing times makes the scheduling problem easy. Most of the time, only one trip will be necessary, and the delivery part of the problem comes down to solving a Traveling Salesman Problem. Therefore, instances were generated to balance the difficulty between the two problems (e.g., we can observe in the optimal solutions several trips with always more than two preparations per batch).

We generated a total of 18 sets of instances. Each set contains 10 instances and is characterized by the number of machines m, the number of jobs n and the mean parameter μ of the normal distribution used to generate the processing times on machine M_1 (see Table 1): The number of machines is equal to 2 or 3, the speed of machine M_i belongs to $\{1, 1.1\}$ if $m = 2$ and to $\{1, 1.05, 1.1\}$ if $m = 3$. n belongs to $\{20, 25, 30, 35, 40\}$. The generation of the processing times follows a normal distribution of mean equal to $\mu \in \{20, 25, 30\}$ time units (minutes) and a standard deviation of 15. Each control takes $p_A = 5$ time units. The release dates r_j are randomly generated in a specified time interval $[0, 0.15\frac{n\mu\sum s_i}{m^2}]$ in order to avoid too much empty space in the scheduling part. Each chemotherapy preparation has a probability equal to 0.15 of being constrained by a stability time st_j. In this case, st_j is set to $p_j + p_A + tt_{0,j} + A$, with A a random value generated using a normal distribution of mean 15 and standard deviation of 5. The due date d_j is derived by adding a randomly drawn value from a normal distribution of mean 35 and standard deviation of 20 to $r_j + p_j + p_A + tt_{0j}$. Each job has a destination service that is randomly selected among 10 sites. These sites are placed at random locations in an area of 12×12 time units, the production service is located at coordinate $(0, 6)$ (traveling times tt_{j_1,j_2} are given by the Euclidean distances).

4.2 Results

To evaluate the efficiency of the Benders decomposition-based heuristic (BDBH), two MILP models have been implemented and tested. The first one is the integer linear programming model (ILP) described in Section 2. The resolution of this model makes it possible to obtain either the optimal solution or an upper bound and a lower bound. To make the comparison equitable, the cuts (25), (34) to (37), (38), (40) and (42) are added to this model to improve computational efficiency, as well as the lower bound described in Section 3.2.2. The second mathematical model aims to find the lower bound of the Benders decomposition. This model, called the Relaxed-Integer Linear Programming model (R-ILP), is therefore composed of two models: the model of the master problem

Instance types			BDBH					ILP		R-ILP	
n	m	μ	#UB	#LB	$\overline{UB-LB}$	#IteBD	CPU	#UB	CPU	#LB	CPU
20	2	20	10	10	20.8	25.5	357.2	10	1557.5	10	507.6
20	2	25	10	10	26.3	48.6	417.1	10	1595.0	10	200.0
20	2	30	10	10	18.5	11.8	174.4	9	1474.8	10	105.5
25	2	20	10	10	31.7	9.7	434.6	9	1800.1	10	1184.7
25	2	25	10	10	28.5	15.4	373.6	9	1683.1	10	686.2
25	2	30	10	10	19.3	9.7	168.8	7	1800.7	10	537.2
30	2	20	10	10	35.4	28.8	863.1	6	1678.3	8	1456.3
30	2	25	10	10	31.4	24.3	828.0	5	1641.0	9	1378.6
30	2	30	10	10	27.4	16.4	684.0	6	1719.7	10	983.2
35	2	20	10	10	34.0	8.7	1516.7	0	-	5	1698.8
35	2	25	10	10	32.8	25.0	1394.1	0	-	3	1757.0
35	2	30	10	10	27.7	15.3	1159.6	2	1800.0	4	1567.5
40	2	20	6	10	47.5	3.2	1800.0	0	-	0	-
40	2	25	6	10	30.5	1.3	1772.1	0	-	1	1800.0
40	2	30	8	10	45.5	5.7	1551.0	1	1800.0	0	-
40	3	20	2	10	60.0	1.1	1800.0	0	-	0	-
40	3	25	2	10	33.5	1.3	1800.0	0	-	0	-
40	3	30	1	10	49.0	1.5	1800.0	0	-	2	1616.0

Table 1: Number of UB and LB and average CPU time.

and the model of the slave problem. For each resolution method, the time limit was set to 30 minutes.

For each method, the first part of the results is given in Table 1. Each line represents a set of 10 instances, described by the three first columns. The next five columns show the number of times an upper bound (#UB) and a lower bound (#LB) are found by BDBH, the mean deviation between the upper bounds and the lower bounds ($\overline{UB-LB}$), the average number of iterations by the Benders decomposition without removing one active Benders cut (#IteBD) and the average CPU time (in seconds). Finally, the last columns give the number of times an upper bound is found by ILP, a lower bound is found by R-ILP, and the average CPU times.

Table 1 shows that BDBH is more efficient than the other methods. It is able to find upper bounds and lower bounds for all instances for $n \leq 35$, whereas the ILP and R-ILP methods encounter difficulties for instances with more than 25 jobs. The quality of these upper bounds and lower bounds are compared in another table. Moreover, BDBH needs less CPU time than the other methods on average. Most of the time, ILP cannot find an optimal solution due to the problem complexity and model size. The CPU time of BDBH and R-ILP are similar for small instances ($n = 20$), but for larger instances, the first method has a CPU time lower than the second one. However BDBH also takes time to compute the upper bounds. From $n = 40$, the resolution of the master problem of BDBH requires too much computation time, and the number of iterations of the heuristics greatly decreased. Finally, the CPU time of BDBH increases when the value of μ decreases. This is explained by relationships between the value of μ and the complexity of the routing part. Indeed, if the values of μ are small, the scheduling part is not really restrictive in comparison with the routing part, which is more CPU consuming. We can observe that the upper bound is rarely equal to the lower bound (only in a few small instances). Since the lower

Instance types			LB_{BDBH} > LB_{ILP}	LB_{BDBH} = LB_{ILP}	LB_{BDBH} < LB_{ILP}
n	m	μ			
20	2	20	0	9	1
20	2	25	0	9	1
20	2	30	2	7	1
25	2	20	3	7	0
25	2	25	1	9	0
25	2	30	3	7	0
30	2	20	4	6	0
30	2	25	3	7	0
30	2	30	3	7	0
35	2	20	6	4	0
35	2	25	10	0	0
35	2	30	7	3	0
40	2	20	10	0	0
40	2	25	7	3	0
40	2	30	9	1	0
40	3	20	4	6	0
40	3	25	6	4	0
40	3	30	3	7	0

Table 2: Comparison of lower bounds of BDBH and ILP

bound is given by relaxing the scheduling part (preemption is allowed), it is logical to obtain a value lower than the upper bound, for which the scheduling part is not relaxed (preemption is not allowed). If the lower bound were often equal to the upper bound (i.e. preemption constraint has no impact), it would mean that the scheduling part is easy to solve for such instances, and only the routing part is difficult. We can also notice that the difference between the upper bound and the lower bound increases when the mean parameter μ (of the normal distribution used to generate the processing times) decreases. This is due to the same reason: When this parameter decreases, the scheduling part becomes easier to solve, and the quality of the lower bound increases.

Tables 2 and 3 compare the quality of the lower bounds between BDBH and ILP, and between BDBH and R-ILP for each type of instance. Column '$LB_{BDBH} > LB_{ILP}$' of Table 2 (respectively Table 3) indicates the number of instances where the lower bound of BDBH is better than the lower bound of ILP (respectively R-ILP). The next column shows the number of instances where both lower bounds are equal and the next column the number of instances where the lower bound of BDBH is worse than the lower bound of ILP (respectively R-ILP). If a method cannot find a lower bound, we consider that the lower bound is equal to 0.

The results presented in Table 2 show that BDBH is clearly able to find better lower bounds than ILP and with shorter computation times (see Table 1). However, when both methods find a lower bound (which does not appear frequently), the difference is not very large. Moreover, for instances with $n \geq 30$ jobs, ILP finds a lower bound and an upper bound a few instances and does not return any lower bound or any solution in most of the cases. The number of cases where the lower bounds are equal is not null because the lower bound is set to 0 when a method cannot find a lower bound. The same interpretation of results can be done for Table 3; BDBH outperforms R-ILP, find-

Instance types			LB_{BDBH} > LB_{R-ILP}	LB_{BDBH} = LB_{R-ILP}	LB_{BDBH} < LB_{R-ILP}
n	m	μ			
20	2	20	0	10	0
20	2	25	0	9	1
20	2	30	0	10	0
25	2	20	2	8	0
25	2	25	0	10	0
25	2	30	0	10	0
30	2	20	2	8	0
30	2	25	1	9	0
30	2	30	0	10	0
35	2	20	4	6	0
35	2	25	8	2	0
35	2	30	5	5	0
40	2	20	10	0	0
40	2	25	6	4	0
40	2	30	9	1	0
40	3	20	4	6	0
40	3	25	6	4	0
40	3	30	3	6	1

Table 3: Comparison of lower bounds of BDBH and R-ILP

ing better lower bounds. There is therefore an interest in using the Benders decomposition method.

The results on the quality of the upper bounds are reported in Table 4. Column $UB_{BDBH} < UB_{ILP}$ gives the number of instances where the upper bound of BDBH is better than the upper bound of ILP. The next column shows the number of instances where both upper bounds are equal and the next column the number of instances where the upper bound of BDBH is worse than the upper bound of ILP. If a method cannot find an upper bound, we consider that the upper bound is equal to the same high value. To assess the gap between the upper bounds found by both methods, we distinguish between two cases:

- instances for which both methods find an upper bound and BDBH dominates; we compute

$$GAP_1 = \frac{UB_{ILP} - UB_{BDBH}}{UB_{ILP}}$$

  and we indicate by #Inst the number of instances where the case occurs.

- instances for which both methods find an upper bound and ILP dominates; we compute

$$GAP_2 = \frac{UB_{BDBH} - UB_{ILP}}{UB_{BDBH}}$$

  and we indicate by #Inst the number of instances where the case occurs.

Table 4 shows that for small instances ($n = 20$), ILP seems to dominate the BDBH. However, this trend is reversed from instances with $n > 20$. Moreover, as the size of instances grows, the gap between the methods increases. This is not surprising because the size of the ILP model is too

Instance types			$UB_{BDBH} < UB_{ILP}$	$UB_{BDBH} = UB_{ILP}$	$UB_{BDBH} > UB_{ILP}$	GAP_1 (#Inst.)	GAP_2 (#Inst.)
n	m	μ					
20	2	20	5	0	5	42.1% (5)	54.1% (5)
20	2	25	3	2	5	33.7% (3)	63.8% (5)
20	2	30	4	1	5	32.4% (3)	54.4% (5)
25	2	20	10	0	0	66.1% (9)	-
25	2	25	10	0	0	69.3% (9)	-
25	2	30	10	0	0	86.5% (7)	-
30	2	20	10	0	0	81.4% (6)	-
30	2	25	10	0	0	88.7% (5)	-
30	2	30	10	0	0	94% (6)	-
35	2	20	10	0	0	-	-
35	2	25	10	0	0	-	-
35	2	30	10	0	0	95.2% (2)	-
40	2	20	6	4	0	-	-
40	2	25	6	4	0	-	-
40	2	30	8	1	1	-	-
40	3	20	2	8	0	-	-
40	3	25	2	8	0	-	-
40	3	30	1	9	0	-	-

Table 4: Upper bound of BDBH and ILP

important for CPLEX solver. For instances with $n \geq 40$, BDBH also has difficulties finding upper bounds; thus both methods are equivalent for these instances. This weakness of BDBH is due to the resolution of the master problem that requires too much CPU time. The number of iterations for this method is thus greatly reduced.

5 Conclusion

This paper presents and formulates an optimization problem from a chemotherapy production and delivery environment, where production and delivery are strongly connected problems. This problem combines a uniform parallel machines scheduling problem and a multi-trip traveling salesman problem with only one salesman. Furthermore, some jobs have a limited lifespan and have to be delivered after the production process starts within a given delay. The objective is to build a schedule and route planning respecting the constraints so that the maximum tardiness of delivery is minimized. To solve this problem, a Benders decomposition is proposed and a Tabu Search heuristic is used to derive an approximate solution. Finally, feasible solutions and lower bounds are obtained. To evaluate the efficiency of the method, two models are developed and solved using a commercial solver. The first one models the whole problem, and the second models a relaxed problem to find lower bounds. Computational experiments are conducted to compare the upper and lower bounds found by the proposed method and the exact resolution of both models. Results show that the Benders decomposition-based heuristic makes it possible to find better lower and upper bounds in reasonable computation times up to medium-sized instances with 40 jobs.

Several research perspectives can be considered. First, the heuristic used to build a feasible solution (the Tabu Search) can be improved: Instead of considering that the routing solution part

is fixed, we can consider that some modifications can be allowed (for example, move a job in a trip to another one in order to adapt the routing solution to the production solution). Another research perspective is to consider several vehicles instead of only one to have a more general model. However, this will increase the complexity of the master problem. Finally, a last research perspective would be to take into account the dynamic aspect of the original problem from the study case (medical drugs production). In the original problem, a job can be canceled during the production process. This case may happen when a patient is not able to receive his treatment for medical reasons. A possible adaptation of the method to a dynamic context can use the rolling horizon method with a sub-horizon for each event (i.e., when data change).

Acknowledgments

This work was supported by the financial support of the ANR ATHENA project, grant ANR-13-BS02-0006 of the French Agence Nationale de la Recherche.

Appendix

References

[Amaro and Barbosa-Póvoa., 2008] Amaro A. C. S. and Barbosa-Póvoa A. P. F. D. (2008). *Supply chain management with optimal scheduling.* Industrial & engineering chemistry research, 47(1), pp. 116-132.

[Amorim, 2013] Amorim, P., Meyr, H., Almeder, C., Almada-Lobo, B. (2013). *Managing perishability in production-distribution planning: A discussion and review.* Flexible Services and Manufacturing Journal, 25 (3), pp. 389-413.

[Belo-Filho et al., 2015] Belo-Filho, M.A.F., Amorim, P., Almada-Lobo, B. (2015). *An adaptive large neighbourhood search for the operational integrated production and distribution problem of perishable products.* International Journal of Production Research, 53 (20), pp. 6040-6058.

[Benders, 1962] Benders J-F. (1962). *Partitioning procedures for solving mixed variables programming problems.* Numerische Mathematik. 4, pp. 238-252.

[Bonfill et al., 2008] Bonfill, A., Espuna, A., Puigjaner, L. (2008). *Decision support framework for coordinated production and transport scheduling in SCM.* Computers and Chemical Engineering, 32(6), pp. 1206?1224.

[Brucker, 2007] Brucker P. (2007). Scheduling algorithms. Fifth edition. Springer.

[Cakici et al., 2015] Cakici, E., Mason, S.J., Geismar, H.N., Fowler, J.W. (2015). *Scheduling parallel machines with single vehicle delivery.* Journal of Heuristics, 20 (5), pp. 511-537.

[Chen, 2004] Chen Z-L. (2004). *Integrated Production and Distribution Operations: Taxonomy, Models, and Review.* Chapter 17 of the book "Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era", edited by D. Simchi-Levi, S.D. Wu, and Z.-J. Shen, Kluwer Academic Publishers.

Table 5: Notation table

Notation	Definition
n	Number of jobs (chemotherapy preparations)
r_j	Release date of job J_j
r'_j	Earliest ready times for delivery of job J_j
p_j	Processing time of job J_j
st_j	Stability time of job J_j
d_j	Delivery due date of job J_j
tt_{j_1,j_2}	Travel time between site j_1 and site j_2
m	Number of uniform parallel machines (pharmacy technicians)
s_i	Production rate of machine M_i
p_A	Control time
M	A large number
$\mathcal{L}$	List of jobs for the delivery from which the routes are defined
$y^i_{j_1,j_2}$	Binary variable to indicate if a job J_{j_1} is scheduled before job J_{j_2} on machine M_i (ILP model)
$x_{j,k}$	Binary variable to indicate if a job J_j is in position k in list $\mathcal{L}$ (ILP model and master problem)
N_k	Binary variable to indicate if a new tour starts in $\mathcal{L}$ with the job at position k (ILP model and master problem)
S_j	Continuous variable of processing starting time of job J_j (ILP model)
D_j	Continuous variable of delivery completion time of job J_j (ILP model and master problem)
t_k	Continuous variable of starting time of the tour delivering the job in position k (ILP model and master problem)
$T_{\max}$	Continuous variable of the maximum tardiness of delivery (ILP model and master problem)
$LB_{T_{\max}}$, $UB_{T_{\max}}$	Lower bound and Upper bound of $T_{\max}$
σ_k	Index of the job in position k in list $\mathcal{L}$ (slave problem)
pd_k	Date at which the job σ_k in position k is ready for analyzing and delivery, considered as data in the slave problem and a variable in the master problem
D_k	Delivery time of the job in position k in list $\mathcal{L}$, considered as data in the slave problem and a variable in the master problem
R_k	Release date of the job delivered in position k in list $\mathcal{L}$, considered as data in the slave problem and a variable in the master problem
P_k	Processing time of the job delivered at position k in list $\mathcal{L}$, considered as data in the slave problem and a variable in the master problem
$pt^l_{i,k}$	Continuous variable of the total amount of time that machine M_i works on the job in position k during the time period $[pd_{l-1}, pd_l]$ (slave problem)
A^l_k	Boolean to indicate if $R_k < pd_l$, considered as a data in the slave problem and a variable in the master problem
u_k, s^l_k, v^l_p, w^l_i	Dual variables of the slave problem

[Chen et al., 2009] Chen, H.-K., Hsueh, C.-F., Chang, M.-S. (2009). *Production scheduling and vehicle routing with time windows for perishable food products.* Computers and Operations Research, 36 (7), pp. 2311-2319.

[Chen, 2010] Chen Z.-L. (2010). *Integrated production and outbound distribution scheduling: Review and extensions.* Operations Research, 58 (1), pp. 130-148.

[Dondo et al., 2011] Dondo R., Mendez C.A. and Cerda J. (2011). *The multi-echelon vehicle routing problem with cross docking in supply chain management.* Computers & Chemical Engineering 35, pp. 3002-3024.

[Dondo and Cerda., 2014] Dondo R. and Cerda J. (2014). *A monolithic approach to vehicle routing and operations scheduling of a cross-dock system with multiple dock doors.* Computers & Chemical Engineering 63, pp. 184-205.

[Erenguc at al., 1999] Erenguç, S.S., Simpson, N.C., Vakharia, A.J. (1999). *Integrated production/distribution planning in supply chains: An invited review.* European Journal of Operational Research, 115 (2), pp. 219-236.

[Fahimnia et al., 2012] Fahimnia, B., Luong, L., Marian, R. (2012). *Genetic algorithm optimisation of an integrated aggregate production-distribution plan in supply chains.* International Journal of Production Research, 50 (1), pp. 81-96.

[Geismar at al., 2008] Geismar, H. N., 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.

[Geoffrion, 1972] Geoffrion AM. (1972). *Generalized benders decomposition.* Journal of optimization theory and applications, 10(4), pp. 237-260.

[Geoffrion, 1974] Geoffrion AM. (1974). *Lagrangean relaxation for integer programming.* In Approaches to integer programming (pp. 82-114). Springer Berlin Heidelberg.

[Glover, 1986] Glover F. (1986). *Future paths for integer programming and links to artificial intelligence.* Computers & Operations Research. 13, pp. 533-549.

[Kergosien and al., 2011] Kergosien Y., Tournamille J-F., Laurence B., Billaut J-C. (2011). *Planning and tracking chemotherapy production for cancer treatment: A performing and integrated solution.* International Journal of Medical Informatics. 80(9), pp. 655-662.

[Lawler and Labetoulle, 1978] Lawler E-L. and Labetoulle J. (1978). *On Preemptive Scheduling of Unrelated Parallel Processors by Linear Programming.* Journal of the association for computing machinery. 25(4), pp. 612-619.

[Lee and Chen, 2001] Lee CY. and Chen Z-L. (2001). *Machine scheduling with transportation considerations.* Journal of Scheduling. 4, pp. 3-24.

[Lee et al., 2014] Lee, J., Kim, B.-I., Johnson, A.L., Lee, K. (2014). *The nuclear medicine production and delivery problem.* European Journal of Operational Research, 236 (2), pp. 461-472.

[Mazier and al., 2010] Mazier A., Billaut J-C., Tournamille J-F. (2010). *Scheduling preparation of doses for a chemotherapy service.* Annals of Operations Research, 178(1): 145-154.

[McDaniel and Devine, 1977] McDaniel D. and Devine M. (1977). *A modified Benders' partitioning algorithm for mixed integer programming.* Management Science, 24, pp. 312-379.

[Sarmiento and Nagi, 1999] Sarmiento, A. M. and Nagi, R. (1999). A review of integrated analysis of production/distribution systems. IIE Transactions, 31(11), pp. 1061-1074.

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

[Van Belle et al., 2012] Van Belle J., Valckenaers P. and Cattrysse D. (2012). *Cross-docking. State of the art.* Omega, 40, pp. 827?845.

[Vidal and Goetschalckx, 1997] Vidal C. and Goetschalckx M. (1997). *Strategic production-distribution models: A critical review with emphasis on global supply chain models.* European Journal of Operational Research, 98, pp.1?18.

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