World Academy of Science, Engineering and Technology
International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering Vol:9, No:7, 2015

Robust Batch Process Scheduling in Pharmaceutical Industries: A Case Study

Tommaso Adamo, Gianpaolo Ghiani, Antonio D. Grieco, Emanuela Guerriero

Abstract—Batch production plants provide a wide range of scheduling problems. In pharmaceutical industries a batch process is usually described by a recipe, consisting of an ordering of tasks to produce the desired product. In this research work we focused on pharmaceutical production processes requiring the culture of a microorganism population (i.e. bacteria, yeasts or antibiotics). Several sources of uncertainty may influence the yield of the culture processes, including (i) low performance and quality of the cultured microorganism population or (ii) microbial contamination. For these reasons, robustness is a valuable property for the considered application context. In particular, a robust schedule will not collapse immediately when a cell of microorganisms has to be thrown away due to a microbial contamination. Indeed, a robust schedule should change locally in small proportions and the overall performance measure (i.e. makespan, lateness) should change a little if at all.
In this research work we formulated a constraint programming optimization (COP) model for the robust planning of antibiotics production. We developed a discrete-time model with a multi-criteria objective, ordering the different criteria and performing a lexicographic optimization. A feasible solution of the proposed COP model is a schedule of a given set of tasks onto available resources. The schedule has to satisfy tasks precedence constraints, resource capacity constraints and time constraints. In particular time constraints model tasks duedates and resource availability time windows constraints. To improve the schedule robustness, we modeled the concept of (a, b) super-solutions, where (a, b) are input parameters of the COP model. An (a, b) super-solution is one in which if a variables (i.e. the completion times of a culture tasks) lose their values (i.e. cultures are contaminated), the solution can be repaired by assigning these variables values with a new values (i.e. the completion times of a backup culture tasks) and at most b other variables (i.e. delaying the completion of at most b other tasks).
The efficiency and applicability of the proposed model is demonstrated by solving instances taken from a real-life pharmaceutical company. Computational results showed that the determined super-solutions are near-optimal.

Keywords—Constraint programming, super-solutions, robust scheduling, batch process, pharmaceutical industries.

I. Introduction

IN pharmaceutical production industry, batch production typically requires the culture of a microorganism population (i.e. bacteria, yeasts or antibiotics). In this application context, the main sources of uncertainty are low performance and quality of the cultured microorganism population as well as microbial contamination, during preparation and fermentation phases. For these reasons, robustness is a key aspect for antibiotic production planning. When a cell of microorganisms has to be thrown away, a robust schedule remains feasible or should change locally in small proportions and the overall performance measure (i.e. makespan, lateness) should change a little if at all.
A classification of problems in terms of plant configurations for pharmaceutical industry is given by [1]. Due to the different plant configurations and process specifications, a wide variety of optimization models, mainly Mixed Integer Linear Programming (MILP) models, have been proposed in the batch process engineering literature. In [2] Maravelias and Grossmann proposed a general hybrid MILP/CP iterative algorithm for the scheduling of multipurpose plants exploiting the strong points of Mathematical and Constraint Programming. Graph based methods usually can be used in order to fit more to the combinatorial nature of scheduling problems. In contrast to the infeasibility problems which may arise with MILP based models, graph based methods never provide infeasible solutions [3]. In this research work we formulated a constraint programming optimization (COP) model for the robust planning of antibiotics production. We developed a discrete-time model with a multi-criteria objective, ordering the different criteria and performing a lexicographic optimization.
The paper is organized as follows. In Section 2 we describe the antibiotics production process in terms of phases and the required resources, taking into account of the priority among activities. In Section 3, we define the problem statements and we give a mathematical formulation for the problem.

T. Adamo, G. Ghiani, A. Grieco and E. Guerriero are with the Department of Engineering for Innovation, University of Salento, Lecce, LE, 73100 Italy {tommaso.adamo, gianpaolo.ghiani, antonio.grieco, emanuela.guerriero}@unisalento.it

II. Antibiotics Production Process

Although most antibiotics are available in nature, such microorganisms are not normally produced in the quantities necessary for large-scale demand. For this reason, each pharmaceutical industry has to design and implement a specific process, typically referred to as fermentation process [4], for each antibiotic. This process (in Fig. 1) involves isolating a desired microorganism, fueling growth of the culture and refining and isolating the final antibiotic product. It is important that sterile conditions be maintained throughout the manufacturing process, because contamination by foreign microbes will ruin the fermentation.

A. Fermentation Process Description

The antibiotic is the output of a production process consisting of four main activities described in the following. In order to better understand the relations between the activities and the resources, we report in Fig. 2 an IDEF0 model. The fermentation process takes place using a culture medium

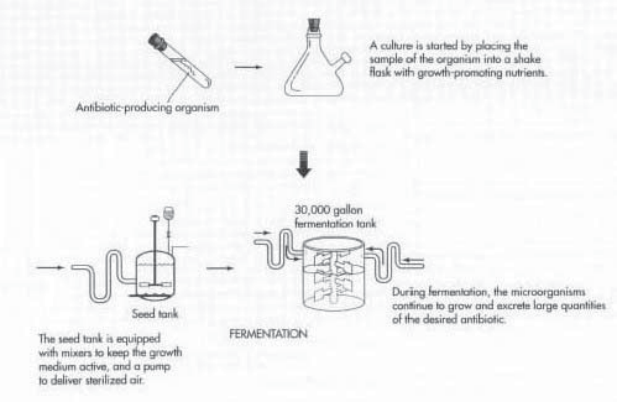

Fig. 1. Fermentation process

and a *charge* of living matter. The culture medium and the charge of living matter are firstly prepared by a *mixer* and a *pre-fermentor*, respectively (*preparation*). The prepared culture medium is firstly transferred to a tank for dilution, then passed through a *sterilizer* and finally loaded into a *fermentor* using a *load collector* (*culture medium inoculum*). Once the culture medium is cooled (*cooling*), the pre-fermented charge is transferred from the *pre-fermentor* to a *fermentor* through a transfer collector (*living charge inoculum*). Then, the growth of the obtained culture is monitored and fueled (*fermentation*) and, after a given amount of time, the antibiotic is isolated (*harvest*).

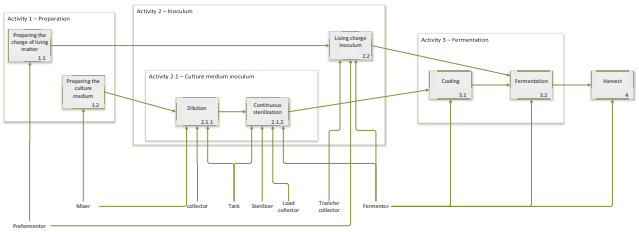

Fig. 2. IDEF0 process model

III. Problem Statements and Mathematical Modeling

We modeled antibiotic production planning as a project scheduling problem as follows. We supposed that the planning horizon is partitioned into N time slots. We denoted with F the set of antibiotic types (i.e. project types), and K the set of resource types (i.e. fermentor, sterilizer, etc.). For each antibiotic type $f \in F$, we defined a work breakdown structure, modeled as a directed graph $G_f = (T_f, P_f)$ where T_f and P_f represent, respectively, the set of tasks and the precedence constraints. For each task $t \in T_f$, we defined a fixed duration s_t and a set of requirements $\Pi_t \subset K$, such that if $k \in \Pi_t$, then the resource type k is required to execute task t. Each task t has a sets of time windows W_t^s (W_t^e), representing the feasible values for its start time (end time). A boolean parameter m_t states if the execution of task t is mandatory or not.

The set R represents the available resources and each $r \in R$ is associated with a resource type $k_r \in K$. For each resource $r \in R$, it is defined a set B_r of forbidden time windows, during which resource r is unavailable (i.e. maintainance). We supposed a non-preemptive operating mode and a unit processing capacity.

A set of products orders O represents the antibiotics demand, in particular each $o \in O$ refers to only one antibiotic type $f_o \in F$. Finally for each order $o \in O$, it is defined an earliest (latest) d_o^{min} (d_o^{max}) ending time for the harvest task. In the following we refer to d_o^{max} as the duedate of order $o \in O$.

A. A Constraint Programming Model

Let alt_{otr} be an interval variable representing the starting and ending times of task t, if assigned to resource $r \in R$, with $o \in O$, $t \in T_{f_o}$ and $k_r \in \Pi_t$. The proposed constraint programming model aims to determine the optimal values of alt variables, minimizing the total lateness with respect to order duedates. In order to model lateness and precedence constraints, we denote with:

- $start(var)$ the starting time for a given interval variable var;
- $end(var)$ the ending time for a given interval variable var;
- $presence(var)$ a boolean value equal to *true* if the interval variable var is scheduled, *false* otherwise.

Let $task_{ot}$ interval variable, representing the start and end processing times of task $t \in T_{f_o}$, $o \in O$. Let δ_o denote the lateness of order o, that is

$$\delta_o = max\left\{0, end(task_{oH_o}) - d_o^{max}\right\},$$

where H_o is the harvest task for the order $o \in O$.

Given an antibiotic type $f \in F$, precedence constraints have been partitioned into three types i.e. $P_f^1 \cup P_f^2 \cup P_f^3 = P_f$. Each type of precedence constraints corresponds to one of the following set of linear constraints.

- `StartAtEnd constraints`: for each $(i,j) \in P_f^1$, task j has to be started exactly τ_{ij} time units after the ending of task i, that is

$$end(i) + \tau_{ij} = start(j).$$

- `StartAfterEnd constraints`: for each $(i,j) \in P_f^2$, the minimum (the maximum) delay between the end of task i and the start of task j is τ_{ij}^{min} (τ_{ij}^{max}):

$$end(i) + \tau_{ij}^{min} \leq start(j) \leq end(i) + \tau_{ij}^{max}$$

- `StartAfterStart constraints`: for each $(i,j) \in P_f^3$, the minimum (the maximum) delay between the start of task i and the start of task j is τ_{ij}^{min} (τ_{ij}^{max}):

$$start(i) + \tau_{ij}^{min} \leq start(j) \leq start(i) + \tau_{ij}^{max}$$

Given a resource $r \in R$, we modeled changeover times as follows. Firstly, we defined a decision variable σ_{rn} denoting

the state of resource $r \in R$ during time slot $n \in N$. In particular, a resource $r \in R$ can be either ready to process an antibiotic type, i.e. $\sigma_{rn} = f \in F$, or not available due to setup operations, i.e. $\sigma_{rn} = nil$. Let $\mathcal{F} = F \cup \{nil\}$ be the domain of σ_{rn}, with $r \in R$ and $n \in N$. The setup cost $M_{r\ell\ell'}$ represents the setup time required if a state transition occurs from $\ell \in \mathcal{F}$ to $\ell' \in \mathcal{F}$ on resource $r \in R$.

In order to generate fault tolerant solutions and ensure robustness, we follow the reformulation approach suggested in [5] to build a super-solution.

A solution to a Constraint Satisfaction Problem is (a, b) *super-solution iff the loss of the values of at most* a *variables can be repaired by assigning other values to these variables, and modifying the assignment of at most* b *other variables.*

In this work we build super-solutions via reformulation with the duplication of some variables. The duplicate variables have the same domain as the original variables, and are linked by the same constraints. The assignment to the original variables is a super-solution, where the repair for each variable is given by its duplicate (see Fig. 3).

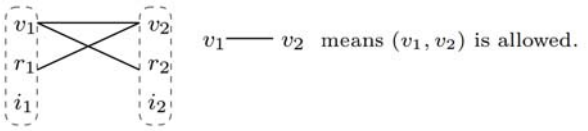

Fig. 3. Reformulation approach

In our approach, firstly we generate a feasible plan that minimizes the total lateness, then we enrich (if it is possible) the schedule of each order with some additional tasks which can be interchanged with the original ones. If there is some kind of breakage or contamination, a repair plan will be easily generated by simply switching tasks. In this case study, the main source of contamination problems is the living charge preparation. Therefore, the duplicate variables model the preparation task of some additional living charges, also called *backup charges*. For each order $o \in O$, we defined the set $Bck_o \subseteq T_{f_o}$ of non-mandatory tasks, representing the production of backup charges.

In the proposed model, the optimization criteria were (in lexicographical order):

1) the minimization of the total lateness, $\min \sum_{o \in O} \delta_o$;
2) the maximization of the minimum number of scheduled backup charges, $\max \min_{o \in O} \sum_{t \in Bck_o} presence(task_{ot})$.

B. Mathematical Formulation

$$\textbf{minimize} \sum_{o \in O} \delta_o, \quad (1)$$

$$\textbf{maximize} \min_{o \in O} \sum_{t \in Bck_o} presence(task_{ot}) \quad (2)$$

s.t.

$$end(task_{oi}) + \tau_{ij} = start(task_{oj}) \quad o \in O, (i,j) \in P^1_{f_o} \quad (3)$$

$$start(task_{oi}) + \tau^{min}_{ij} \leq start(task_{oj}) \leq start(task_{oi}) + \tau^{max}_{ij} \quad o \in O, (i,j) \in P^2_{f_o} \quad (4)$$

$$start(task_{oi}) + \tau^{min}_{ij} \leq start(task_{oj}) \leq start(task_{oi}) + \tau^{max}_{ij} \quad o \in O, (i,j) \in P^3_{f_o} \quad (5)$$

$$d^{min}_o \leq end(task_{oH_o}) \leq \delta_o + d^{max}_o, \quad o \in O \quad (6)$$

$$alternative(task_{ot}, \bigcup_{r \in R: k_r \in \Pi_t} alt_{otr}), \quad o \in O, t \in T_{f_o} \quad (7)$$

$$noOverlap(\bigcup_{\substack{o \in O, t \in T_{f_o}: \\ k_r \in \Pi_t}} alt_{otr}), \quad r \in R \quad (8)$$

$$a \leq start(task_{ot}) \leq b, \quad o \in O, t \in T_{f_o}, (a,b) \in W^s_t \quad (9)$$

$$a \leq end(task_{ot}) \leq b, \quad o \in O, t \in T_{f_o}, (a,b) \in W^e_t \quad (10)$$

$$start(task_{ot}) \in N,\ end(task_{ot}) \in N, \quad o \in O, t \in T_{f_o} \quad (11)$$

$$end(task_{ot}) - start(task_{ot}) = s_t, \quad o \in O, t \in T_{f_o} \quad (12)$$

$$m_t \Rightarrow presence(task_{ot}), \quad o \in O, t \in T_{f_o} \quad (13)$$

$$end(alt_{otr}) < a \ \vee \ start(alt_{otr}) > b, \quad o \in O, t \in T_{f_o}, r \in R, (a,b) \in B_r : k_r \in \Pi_t \quad (14)$$

$$\sigma_{rn} = f_o, \quad o \in O, t \in T_{f_o}, r \in R$$

$$n \in [start(alt_{otr}), end(alt_{otr})] \cap N : k_r \in \Pi_t \quad (15)$$

$$\sigma_{rn} \neq \sigma_{rn'} \Rightarrow n' - n \geq M_{r\sigma_{rn}\sigma_{rn'}}, \quad n, n' \in N : n' > n, r \in R \quad (16)$$

$$\sigma_{rn} \in \mathcal{F}, \quad r \in R, n \in N \quad (17)$$

$$\sigma_{r \min\{N\}} = \sigma^0_r, \quad r \in R \quad (18)$$

$$\delta_o \in \left[0, \delta^{max}_{f_o}\right] \cap \mathbb{N}, \quad o \in O \quad (19)$$

First objective function (1) is the total lateness, whilst second objective function (2) is the robustness. Constraints (3), (4) and (5) are the precedence constraints described in the previous section. For each order $o \in O$, constraint (6) states that the end of the harvest task H_o has to occur in $[d^{min}_o, d^{max}_o + \delta_o]$. Constraint (7) models the relationship between the *alt* and the *task* variables. Constraint (8) represents the unit processing capacity for each resource $r \in R$. Constraints (9) and (10) model time windows W^s_t and W^e_t. Constraint (11) defines the domain of the *task* variables, while constraint (12) states their duration. Constraint (13) imposes the schedule of mandatory operations. For each resource $r \in R$, constraint (14) models the forbidden time windows B_r. Constraints (15) and (16) model state transitions applying the changeover times. We consider by convention that the changeover time from or to the *nil* status is equal to zero. Constraints (17) and (18) state respectively the domain and the initial value of states variables. Constraint (19) states an upper bound for the lateness.

IV. Computational Results

The efficiency and applicability of the proposed model was demonstrated by solving instances taken from a real-life pharmaceutical company. In Fig. 4 and 5 we reported the Gantt charts corresponding to the output of one instance. We used one different color for each $f \in F$. On the left, the labels refer to resources $r \in R$. In particular, labels F_k refer to fermentors that are the main bottlenecks (see Fig. 4).

When our research team began to deal with this decision problem, the production planning was determined by applying an earliest duedate heuristic (EDD). The production manager was wondering if their monthly throughput could be increased.

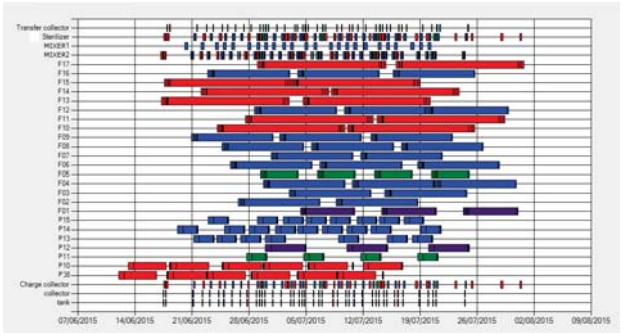

Fig. 4. Gantt chart: minimizing total lateness

The computational results pointed out that throughput can be increased only by adding one more fermentor. It is worth noting that adding one fermentor is a strategic decision since it requires huge investments (about one million dollars).

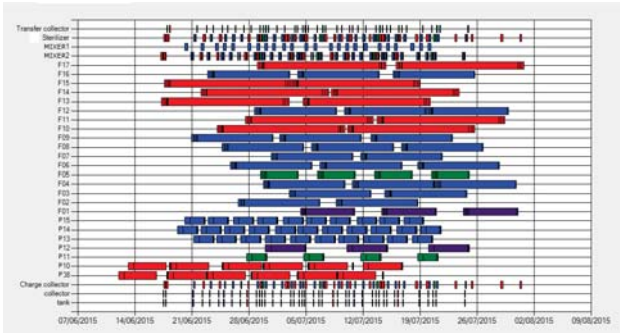

Fig. 5. Gantt chart: minimizing lateness and maximizing robustness

As far as robustness is concerned, Fig. 5 reports the Gantt chart obtained when we optimized in lexicographical order lateness and robustness: seven backup charges were added with respect to solution of Fig. 4. In general, our computational results showed that the proposed model was equivalent in terms of robustness, but provided a gain on lateness of 20% on average with respect to EDD heuristic. Finally, the EDD heuristic has a granularity of a day, whilst our model provides hourly details on the operational activities.

V. Conclusions

This work presents a constraint programming framework for the antibiotics production planning problem. We proposed a model for minimizing in lexicographical order total lateness and robustness, taking into account precedence, time and resource capacity constraints. Instances from a real-life pharmaceutical company showed a gain on lateness of 20% on average. The adoption of an automated tool for production scheduling is a way to improve productivity and to promote integration among strategic and operational decision making layers. This kind of vertical integration is a key step of Industry 4.0 [6]. In particular, this research work showed that robust constraint programming is an effective and efficient tool for planning industrial production of microorganism cultures.

References

[1] C. T. Maravelias and I. E. Grossmann, "Using milp and cp for the scheduling of batch chemical processes," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Springer, 2004, pp. 1–20.

[2] T. Maravelias and I. E. Grossmann, "A hybrid milp/cp decomposition approach for the scheduling of batch plants," in *Proceedings of CP-AI-OR*, 2003.

[3] M. Hegyháti, T. Majozi, T. Holczinger, and F. Friedler, "Practical infeasibility of cross-transfer in batch plants with complex recipes: S-graph vs milp methods," *Chemical Engineering Science*, vol. 64, no. 3, pp. 605–610, 2009.

[4] W. T. Hess, A. Kurtz, and D. Stanton, "Kirk-othmer encyclopedia of chemical technology," *John Wiley & Sons Ltd., New York*, 1995.

[5] E. Hebrard, B. Hnich, and T. Walsh, "Super solutions in constraint programming," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Springer, 2004, pp. 157–172.

[6] "Project of the future: Industry 4.0," http://www.bmbf.de/en/19955.php.