Clustering-based solution approach for a capacitated lot-sizing problem on parallel machines with sequence-dependent setups

François Larroche, Odile Bellenguez, Guillaume Massonnet

Clustering-based solution approach for a capacitated lot-sizing problem on parallel machines with sequence-dependent setups

François Larroche[a, b], Odile Bellenguez[a] and Guillaume Massonnet[a]

[a]IMT Atlantique, LS2N, La Chantrerie, 4 rue Alfred Kastler, 44307 Nantes, France
[b]VIF, 10 Rue de Bretagne, 44240 La Chapelle-sur-Erdre, France

ABSTRACT
This paper studies an industrial lot-sizing and scheduling problem coming from the food-industry that extends the multi-item capacitated lot-sizing and includes lost sales, overtimes, safety stock and non uniform sequence-dependent setups on parallel machines. We introduce two different formulations and adapt the well-known Relax-and-Fix and Fix-and-Optimize heuristics in order to quickly obtain feasible solutions on large industrial instances. The complexity of our problem prevents the procedure to obtain good solutions within the time allocated by practitioners on real-life cases, hence we propose to use a clustering approach to approximate the sequence-dependent setup times. The resulting problem is significantly smaller to solve and experimental results suggest that this transformation effectively improves the solutions found on industrial instances. In particular, the combination of this clustering method and Relax-and-Fix and Fix-and-Optimize procedure turns out to be a promising approach to obtain good solutions in the given time-limit.

KEYWORDS
Lot-sizing; sequence-dependent setups; parallel machines; clustering

1. Introduction

In this paper, we focus on production problems that arise in practical cases from the food industry and involve the manufacturing of several items on multiple production lines. The main goal of the industrial application behind this study is to reach "good" feasible solutions in a short computational time, providing the planners with a decision support tool that they can use on a daily basis. Unfortunately, those problems generally combine multiple lot-sizing and scheduling constraints, which makes them difficult to handle using a straightforward Mixed Integer Linear Programming (MILP) modeling. In particular, when the setup times between different types of items heavily depend on the production sequence, such approaches lead to large mathematical formulations that turn out to be intractable by commercial solvers. Thus, fast and reliable dedicated methods are necessary in order to meet practitioners' requirements.

We call this problem the *multi-item capacitated lot-sizing problem with lost sales, safety stock, overtimes, and sequence-dependent setups on parallel machines*, abbreviated CLSSD-PM in the remainder of this paper. It extends many existing lot-sizing models that have been developed in the production planning literature following the

Supported by VIF Software
CONTACT François Larroche Email: francois.larroche@imt-atlantique.fr

work of Wagner and Whitin (1958). The ever-growing need to propose customized models that allow to precisely describe and solve a wide range of industrial production processes has shifted the interest of manufacturers towards different generalizations of the original problem, see for instance Jans and Degraeve (2008) for a comprehensive survey of these applications. The version that we consider in this paper regroups some of these well-known extensions, along with new ones specific to our industrial case. In particular, it includes several aspects related to the Capacitated Lot Sizing Problem (CLSP) and the Asymmetric Travelling Salesman Problem (ATSP). Both problems are proven NP-hard (see Florian, Lenstra, and Rinnooy Kan (1980) and Karp (1972), respectively), hence it is straightforward that the CLSSD-PM is itself NP-hard.

The capacity constraints that appear in the practical applications we consider are similar to the ones found in the CLSP, see Karimi, Fatemi Ghomi, and Wilson (2003) or Quadt and Kuhn (2008) for reviews of extensions and solution approaches. Lost sales is another aspect of the CLSSD-PM that frequently appears in the lot-sizing literature. In particular, it is studied by Absi and Kedad-Sidhoum (2008), who introduce new classes of valid inequalities. In Absi, Detienne, and Dauzère-Pérès (2013), a Lagrangian heuristic is developed and provides good results for the CLSP with lost sales. On the other hand, the safety stock is seldom considered in the deterministic production and inventory literature. Loparic, Pochet, and Wolsey (2001) define the safety stock as a lower bound on the number of units that must be held in the inventory in each period. However, in many real-life cases, it is mainly seen as a target inventory level, set to mitigate the risk of shortage rather than an hard constraint. Following this approach, Absi and Kedad-Sidhoum (2009) choose to penalize missing units from the safety stock in the objective function and develop a Lagrangian heuristic approach to solve the problem.

The combination of production planning and scheduling problems is relevant in practice, since setup times are often sequence-dependent. This observation has motivated the introduction of integrated models generally referred to as *lot-sizing and scheduling problems*, that consider both levels of decision to properly approximate the capacity consumption in each time period. For instance, Gicquel et al. (2009) present a discrete lot-sizing and scheduling problem (DLSP) based on a small bucket formulation, i.e. the production in each period is limited to only one type of item, with the additional constraint that any production must be at full capacity. In contrast, the so-called big bucket formulations consider that time periods are long enough to produce multiple types of items. Fleischmann and Meyr (1997) are among the first to study a sequence-dependent version of this problem, called the *General Lot-sizing and Scheduling Problem* (GLSP), and consider two time structures, namely a set of fixed periods (big buckets) that contain sequences of smaller variable production periods (small buckets). For a state of art of these different variants of the lot-sizing and scheduling problem, the reader can refer to the extensive reviews of Guimarães, Klabjan, and Almada-Lobo (2014) and Copil et al. (2017).

In most cases, complex lot-sizing problems with capacity restriction, sequence-dependent (SD) setups and/or multiple machines are tackled using decomposition and constructive heuristics to obtain feasible solutions. Almada-Lobo et al. (2007) propose an iterative heuristic to solve the capacitated lot-sizing problem with sequence-dependent setup times and costs (CLSD) and compare its result with a lower bound from a strengthened formulation. The well-known *Relax-and-Fix* is also a popular decomposition heuristic that usually offers a good trade-off between solution quality and runtime. It is frequently combined with the local search heuristic *Fix-and-Optimize* to improve the solutions. We refer the reader to Absi and Kedad-Sidhoum (2007) for

a thorough presentation of this class of methods, including an extensive parameter analysis. Toso, Morabito, and Clark (2009) apply this type of procedure to the GLSP, while Clark, Morabito, and Toso (2010) propose a solution procedure based on subtour elimination and a patching heuristic. Lang and Shen (2011) use a Fix-and-Optimize heuristic to address the single machine version of CLSD including overtimes. Fiorotto, Neyra, and Araujo (2019) apply a Relax-and-Fix and Fix-and-Optimize procedure to obtain feasible solutions for difficult instances of a single machine problem with sequence-independent setup times including crossover (i.e. start a setup and end it in the next period) and carryover (i.e. continue the production of an item from one period to the next without incurring a new setup).

In addition, problems that include multiple resources are divided into two broad categories: identical (Beraldi et al. (2008)) and non-identical (Kaczmarczyk (2013)) parallel machines. The problem CLSSD-PM that we address considers non-identical production machines. It extends the case of sequence-dependent setup times and costs on parallel machines presented in James and Almada-Lobo (2011), for which the authors develop new MIP-based neighborhood search heuristics and compare them to other solution procedures. Several methods have been experimented for similar frameworks. Almada-Lobo et al. (2010) consider a version of the problem in which a given machine can only produce one type of item per period and propose to combine the addition of valid inequalities with a Lagrangian heuristic to obtain good solutions. Mateus et al. (2010) solve an integrated version of the problem with an iterative procedure, while Xiao et al. (2013) develop an original Fix-and-Optimize algorithm to address a problem from the semiconductor manufacturing industry. Other works such as Fiorotto and de Araujo (2014) and Xiao et al. (2015) rely on Lagrangian decomposition heuristics to deal with parallel machines problems. More recently, Armas and Laguna (2020) study a similar problem, from the pipe-insulation industry, and propose a solution procedure that runs in two phases: In the first one, a simplified version of the problem with sequence-independent setups is solved, before re-scheduling the production using a post-processing heuristic in the second phase.

The remainder of this paper is organized as follows: Section 2 describes the notations and assumptions of the CLSSD-PM problem, including a presentation of the industrial context that explains the introduction of production sequence. In Section 3, we propose two different mathematical formulations for our problem, respectively referred to as the aggregate and the facility location ones. In Section 4, we succinctly presents the so-called Relax-and-Fix and Fix-and-Optimize heuristics adapted to the CLSSD-PM problem in order to obtain feasible solutions in a short computational time on large instances. Then we develop an approach based on a clustering procedure to approximate the sequence-dependent setup times. The created clusters are used to reduce the problem size and obtain good solutions within a limited computational time. In the following, we introduce a two-phases approach: The first one aims at finding a solution to the modified clustered-problem, which is then used in the second one to build optimal production sequences for each machine in each period. This approach ensures that any solution to the modified problem remains feasible in the original framework. Finally, Section 5 presents a sensitivity analysis for parameters tuning and computational results apply on instances inspired by practical data sets from our industrial partners. Numerical results suggest that even for medium size instances, our approximation of the setup times enables us to quickly find better solutions to the CLSSD-PM, regardless to the type of formulation used.

Table 1. Sets and constants
N: Number of items
$\mathcal{N} = \{1, \ldots, N\}$
M: Number of machines
$\mathcal{M} = \{1, \ldots, M\}$
T: Number of periods
$\mathcal{T} = \{1, \ldots, T\}$

2. Problem description

2.1. *Notations and assumptions*

The problem CLSSD-PM is an extension of the CLSP with lost sales and safety stock on (unrelated) parallel machines. The goal is to plan the production of N different items over a discrete finite horizon of T periods, each one considered as a big bucket, on M non-identical parallel machines (or other production resources). For conciseness, we denote $\mathcal{N}$, $\mathcal{M}$ and $\mathcal{T}$ the set of items, machines and periods, respectively (see Table 1). In each period $t \in \mathcal{T}$, there is a deterministic demand d_t^i for each item $i \in \mathcal{N}$, which is either lost or satisfied from on-hand inventory. For notation convenience, we also let $D_{tt'}^i = \sum_{u=t+1}^{t'} d_u^i$ be the cumulative demand from period $t+1$ to t' for all $t < t'$. While it is often advised to minimize the physical stock at all time, production managers usually set a safety stock S_t^i for each item i and period t in order to absorb uncertain perturbations of the production process (e.g. machine failures). For all $i \in \mathcal{N}$ and $m \in \mathcal{M}$, we let τ_m^i represent the per-unit production time of item i on machine m. Each machine $m \in \mathcal{M}$ has a time-dependent production capacity C_{mt}, which corresponds to the expected production time necessary in period t, estimated beforehand by the S&OP (Sales and Operations Planning). However, the planner can choose to exceed this planned capacity for machine m up to a hard limit $\overline{C}_{mt}$ which is the "true" available time capacity in period t. In that case, $\overline{C}_{mt} - C_{mt}$ corresponds to the maximum overtime allowed. Finally, we also impose that such a production is greater than a minimum production quantity q_{min}^i. For a given machine m, the setup time required to switch from one type of item i to another type $j \neq i$ depends on the pair (i, j) and is denoted λ_m^{ij}. We assume that the sequence-dependent setup times satisfy the triangle inequality.

All the above operations induce costs. Specifically, producing one unit of item i in period t generates a per-unit production cost p_{mt}^i. In addition, using machine m for production or setup during period t incurs a cost c_{mt} per time-unit of usage. Whenever an overtime is requested, each extra unit of time needed is charged at an additional cost $\bar{c}_{mt}$. Recall that the production planner aims at keeping the stock of each final product as close as possible to its target inventory level. We penalize any deviation from this objective with a holding cost h_t^{i+} that applies to each excess unit of item i over the safety stock S_t^i at the end of period t. On the other hand, failing to meet this safety stock requirement also induces a penalty cost h_t^{i-} for each missing unit. The capacity restrictions that apply to the production process may also lead to some unsatisfied demands. Every unmet unit of demand for item i induces a shortage cost l_t^i (lost-sales) in period t. The objective is to minimize the sum of inventory holding costs, shortage costs, production costs and line usage costs. The different input data are summarized in Table 2.

Table 2. Models parameters

d_t^i	Demand for item i in period t
C_{mt}	Capacity (time) available on machine m in period t
$\overline{C}_{mt}$	Total capacity available on machine m in period t, including possible overtimes
τ_m^i	Time necessary to produce one unit of item i on machine m
q_{min}^i	Minimum production quantity of item i
S_t^i	Safety stock for item i in period t
λ_m^{ik}	Changeover time from item i to item k on machine m
h_t^{i+}	Per-unit holding cost for excess inventory of item i in period t
h_t^{i-}	Per-unit penalty cost for item i in period t applied to inventory shortfall under S_t^i
l_t^i	Per-unit lost-sales cost for item i in period t
p_{mt}^i	Per-unit production cost for item i on machine m in period t
c_{mt}	Cost of usage time for machine m in period t, per time unit
$\bar{c}_{mt}$	Cost of overtime for machine m in period t, per time unit

2.2. *Production sequence*

We focus on a version of the problem in which the periods are considered as independent. Machines are reset between two consecutive periods, hence no carryover is considered and each machine is regarded as idle at the beginning of each period. Moreover, we assume that the first setup is performed outside of the time slot available for production in each period. This assumption is rare in related works on lot-sizing and scheduling. To the best of our knowledge, it only appears in Clark, Morabito, and Toso (2010). In our case, this is in line with many food-factory practices, in which a team is assigned to the setup of machines before the actual production period begins. We model this idle state of a machine using a virtual item indexed 0, which is always considered as active, i.e. in production, at the beginning of a period. For notational convenience we also let $\mathcal{N}_0 = \mathcal{N} \cup \{0\}$ and artificially define null setup times between item 0 and any other item $i \in \mathcal{N}$, i.e. $\lambda_m^{0i} = 0$ for all $m \in \mathcal{M}$.

One of the main challenges of the CLSSD-PM problem lies in the influence of the production sequence on the setup times. The so-called sequence-dependent setups extension of lot-sizing problem reconciles production planning and scheduling problems, in the sense that the two levels of decision are integrated. In the food-industry, as in many other fields of application, the sequence-dependent setups are determined on the basis of some items characteristics. One can think of the transition from non-organic to organic products which requires a long clean-up to meet the sanitary standards, while clean-up time is negligible in the opposite sequence, as an example of such features likely to influence the setup times. In the CLSD case, the main works that deal with sequence-dependent setups are related to the ATSP problem. Indeed once the production mix is determined in a given period, the optimal production sequence corresponds to the optimal tour for an ATSP on the selected subset of items. Figure 1 presents two different sequences for a problem with 4 items on a single machine. The two solutions are equivalent in terms of quantity produced in the period considered, but differ in their respective cost due to a longer production overtime in the upper sequence. As in the ATSP, the difficult part of this problem largely comes from the subtour elimination constraints. For a quick representation of the different subtour configurations, see Almada-Lobo et al. (2007). Note that in the case of the CLSSD-PM problem, this issue becomes even more significant with the multiplication of production resources, each requiring to solve an ATSP problem for any subset of items affected to it.

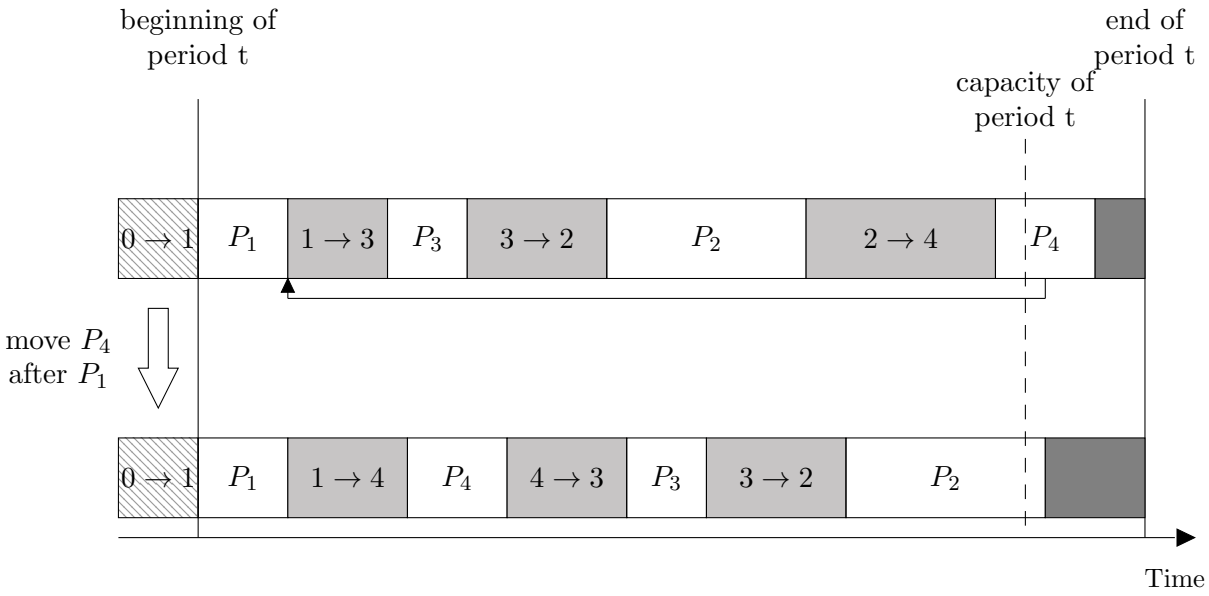


Figure 1. Illustrative example of two production sequences

3. Mathematical formulations

In the remainder of this section, we present two different formulations for the CLSSD-PM. For conciseness, we first introduce decision variables and constraints that are shared by both formulations, before giving a more detailed description for each of them. In all of our models, we use the binary variables $w_{mt}^{ij} \in \{0,1\}$ to indicate if item i is the immediate predecessor of item j in the production sequence on machine m in period t. We also use U_{mt} and $O_{mt} \in \mathbb{R}_+$ to represent the total production plus setup time and the total overtime of machine m in period t, respectively.

Several constraints, relative to capacity restrictions and subtours elimination, are present in all of our formulations, namely :

– Enforce the total production time of machine m in period t to be lower than the capacity allocated to it, plus the possible overtime fixed by the planner:

$$U_{mt} \leq C_{mt} + O_{mt} \quad \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{1}$$

– Prevent the overtime to exceed the difference between the physical (hard) capacity and the planned capacity in a given period:

$$O_{mt} \leq \overline{C}_{mt} - C_{mt} \quad \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{2}$$

– Impose that an item can only be setup if there exists a setup from the neutral state to any real item:

$$\sum_{i \in \mathcal{N}} w_{mt}^{0i} \geq \sum_{i \in \mathcal{N}_0} w_{mt}^{ij} \quad \forall j \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{3}$$

– Allow a setup from an active item only if it belongs to the production sequence

of the period:

$$\sum_{j \in \mathcal{N}_0} w_{mt}^{ji} \geq \sum_{j \in \mathcal{N}} w_{mt}^{ij} \quad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{4}$$

– Forbid more than one setup per item:

$$\sum_{j \in \mathcal{N}} w_{mt}^{ij} \leq 1 \quad \forall i \in \mathcal{N}_0, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{5}$$

– Prevent disconnected subtours: The goal is to ensure that in each period t the sequence variables w_{mt}^{ij} defines a single cycle that starts and ends in the neutral item 0.

Multiple options are available from the literature to satisfy this property: One of the most frequently seen relies on the so-called MTZ constraints, see e.g. James and Almada-Lobo (2011). More recently, Guimarães, Klabjan, and Almada-Lobo (2014) introduces the *Single Commodity Flow* (SCF) constraints and provides numerical evidences of their superior performances compared to MTZ.

Formulations based on the SCF constraints turned out to be more efficient in our numerical experiments, hence we focus on the latter in the remainder of the paper. For all $i \in \mathcal{N}_0$,$j \in \mathcal{N}$ $m \in \mathcal{M}$ and $t \in \mathcal{T}$, we introduce flow variables $f_{mt}^{ij} \in \mathbb{R}_+$ that represent the flow of one commodity from item i to item j on machine m in period t. The following set of constraints prevents the subtours in the sequence :

$$\sum_{j \in \mathcal{N}} f_{mt}^{0j} = \sum_{i \in \mathcal{N}_0} \sum_{j \in \mathcal{N}} w_{mt}^{ij} \qquad \forall m \in \mathcal{M},\ \forall t \in \mathcal{T} \tag{6}$$

$$\sum_{i \in \mathcal{N}_0} f_{mt}^{ij} = \sum_{i \in \mathcal{N}_0} w_{mt}^{ij} + \sum_{i \in \mathcal{N}} f_{mt}^{ji}\ \forall j \in \mathcal{N},\ \forall m \in \mathcal{M},\ \forall t \in \mathcal{T} \tag{7}$$

$$0 \leq f_{mt}^{ij} \leq N w_{mt}^{ij} \qquad \forall i \in \mathcal{N}_0, \forall j \in \mathcal{N},\ \forall m \in \mathcal{M},\ \forall t \in \mathcal{T} \tag{8}$$

Constraints (6) ensure that for each period and machine, the total commodity flow from the source corresponds to the number of items produced in the period on each machine. Constraints (7) define the flow balance. Constraints (8) impose an upper bound on the amount of flow.

– Define the domain of every variable :

$$w_{mt}^{ij} \in \{0,1\}\ \forall i \in \mathcal{N}_0, \forall j \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{9}$$

$$U_{mt}, O_{mt} \geq 0 \qquad \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{10}$$

We are now ready to introduce the two mathematical formulations for the CLSSD-PM.

3.1. *Aggregate formulation MIP-AGG*

We first present an aggregate formulation, which is the most common to describe lot-sizing problems. Specific decision variables for this model are summarized below:

$x^i_{mt} \in \mathbb{R}_+$	Quantity of item i produced in period t on machine m
$L^i_t \in \mathbb{R}_+$	Quantity of lost sales for item i in period t
$I^i_t \in \mathbb{R}_+$	Inventory of item i on hand at the end of period t
$I^{i+}_t \in \mathbb{R}_+$	Overstock (based on safety stock value) of item i at the end of period t
$I^{i-}_t \in \mathbb{R}_+$	Safety stock deficit of item i at the end of period t
$z^i_t \in \{0,1\}$	Binary variable equals to 1 if the stock of item i is null at the end of period t

The aggregate formulation, inspired by Clark, Morabito, and Toso (2010), is expressed with the following MIP:

$$\min \sum_{t\in\mathcal{T}}\sum_{m\in\mathcal{M}} (c_{mt}U_{mt} + \overline{c}_{mt}O_{mt}) + \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}} \left(h^{i+}_t I^{i+}_t + h^{i-}_t I^{i-}_t + l^i_t L^i_t + \sum_{m\in\mathcal{M}} p^i_{mt} x^i_{mt} \right) \tag{11}$$

$$\text{s.t. } U_{mt} = \sum_{i\in\mathcal{N}} (\tau^i_m x^i_{mt} + \sum_{j\in\mathcal{N}} \lambda^{ji}_m w^{ji}_{mt}) \quad \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{12}$$

$$I^i_t = S^i_t + I^{i+}_t - I^{i-}_t \quad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{13}$$

$$I^i_t = I^i_{t-1} + \sum_{m\in\mathcal{M}} x^i_{mt} + L^i_t - d^i_t \quad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{14}$$

$$I^i_t \leq D^i_{tT}(1 - z^i_t) \quad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{15}$$

$$L^i_t \leq z^i_t d^i_t \quad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{16}$$

$$x^i_{mt} \leq D^i_{t-1,T} \sum_{j\in\mathcal{N}_0} w^{ji}_{mt} \quad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{17}$$

$$x^i_{mt} \geq q^i_{\min} \sum_{j\in\mathcal{N}_0} w^{ji}_{mt} \quad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{18}$$

Constraints (1)-(10)

$$I^{i+}_t \leq D^i_{tT} - S^i_t \quad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{19}$$

$$I^{i-}_t \leq S^i_t \quad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{20}$$

$$L^i_t, I^i_t, I^{i+}_t, I^{i-}_t \geq 0 \quad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{21}$$

$$x^i_{mt} \geq 0 \quad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{22}$$

Constraints (12) define the total working time of each machine which is equal to the production time and the setup times. Constraints (13) defines the inventory on hand for item i at the end of period t, which is equal to the safety stock value plus the overstock minus the deficit. Constraints (14) are the inventory flow conservation equations through the planning time horizon. Constraints (15) and (16) ensure that lost sales for item i occur only when the corresponding stock is null. Constraints (17) use the cumulative demand over the remainder of the horizon as an upper bound on the quantity of each item produced on each machine in a given period. Constraints (18) force the production to be higher than its minimum value. Finally constraints (19) and (22) define the domain of specific variables.

3.2. *Facility location formulation MIP-FL*

The second formulation we present is inspired by the facility location problem. This idea was originally developed in Krarup and Bilde (1977) for the uncapacitated lot-sizing problem and adapted later to several of its extensions. It requires to define new variables $x^{it}_{ms} \in \mathbb{R}_+$ that represent the quantity of item i produced in period s on machine m to satisfy a demand in period t. In this formulation, linear costs are written using new parameters denoted by H^{it}_{ms} that aggregate all the costs induced by the production in period s till the demand satisfaction in period t, i.e. the production cost and holding costs of the corresponding units. We introduce additional notations to define the value of H^{it}_{ms}. For a given demand i and period t, we let $\theta^i_t \leq t$ be the first period in which the safety stock contains units of item i that serve demand d^i_t. Note that since θ^i_t is uniquely defined for all pair (i,t), we implicitly assume that the safety stock is defined in such a way that it covers the demand of an integer number of future periods. This effectively turns the safety stock into a safety cover, which is the case in practice. Using this new notations, one can express the inventory costs in relation with the safety stock using only the production period s and the consumption period t. Formally, we define parameters H^{it}_{ms} as follow:

$$H^{it}_{ms} = p^i_{ms} - l^i_t + \sum_{u=s}^{\theta^i_t-1} h^{i+}_u - \sum_{u=\max\{\theta^i_t,s\}}^{t-1} h^{i-}_u \tag{23}$$

Note that the coefficients in front of the lost-sales and safety stock penalty costs in (23) are negative. This is due to the fact that they both represent a deficit to a target level (demand and safety stock, respectively), which is *reduced* through the production of units of item i. In contrast, production and excess holding costs both *increase* with the production of new units, which explains their positive coefficients in (23). Therefore, one simply needs to add the constant $\sum_{i\in\mathcal{N}}\sum_{t\in\mathcal{T}}\left(l^i_t + \sum_{u=\theta^i_t}^{t-1} h^{i-}_u\right) d^i_t$ to compare the solutions values to the ones obtained with the MIP-AGG formulation.

We define the MIP-FL formulation as follows :

$$\min \sum_{s\in\mathcal{T}}\sum_{m\in\mathcal{M}}\left(c_{ms}U_{ms}+\overline{c}_{ms}O_{ms}+\sum_{i\in\mathcal{N}}\sum_{t=s}^{T}H_{ms}^{it}x_{ms}^{it}\right) \tag{24}$$

$$\text{s.t} \quad U_{ms}=\sum_{i\in\mathcal{N}}\left(\tau_m^i\sum_{t=s}^{T}x_{ms}^{it}+\sum_{j\in\mathcal{N}}\lambda_m^{ij}w_{ms}^{ij}\right) \quad \forall m\in\mathcal{M},\forall s\in\mathcal{T} \tag{25}$$

$$x_{ms}^{it}\leq d_t^i\sum_{j\in\mathcal{N}_0}w_{ms}^{ji} \quad \forall i\in\mathcal{N},\forall m\in\mathcal{M},\forall s\in\mathcal{T} \tag{26}$$

$$\sum_{m\in\mathcal{M}}\sum_{s=1}^{t}x_{ms}^{it}\leq d_t^i \quad \forall i\in\mathcal{N},\forall t\in\mathcal{T} \tag{27}$$

$$\sum_{t=s}^{T}x_{ms}^{it}\geq q_{\min}^i\sum_{j\in\mathcal{N}_0}w_{ms}^{ji} \quad \forall i\in\mathcal{N},\forall m\in\mathcal{M},\forall s\in\mathcal{T} \tag{28}$$

$$\text{Constraints (1)-(10)}$$

$$x_{ms}^{it}\geq 0 \quad \forall i\in\mathcal{N},\forall m\in\mathcal{M},\forall t\in\mathcal{T},\forall s=1,\ldots,t \tag{29}$$

The objective function (24) is still the minimization of the total cost incurred. Constraints (25) is just a reformulation of (12) with the new variables x_{ms}^{it}. Constraints (26) impose that an item can only be produced if the corresponding machine is setup for it. Constraints (27) ensure that it is impossible to produce more that the demand in period t from productions up to that period. Constraints (28) set the lower bound on production quantities. Finally, constraints (29) define the specific variables.

4. Solution approach

The industrial specifications relative to this problem impose to solve the MIP presented in the previous section in a short execution time. Unfortunately, both formulations are too large and complex to quickly obtain good (or even feasible) solutions on any realistic data set with any of the existing commercial solvers. In this section, we propose alternative solution approaches to tackle this problem. A classical approach in such cases is to use heuristics to obtain good, albeit often suboptimal, solutions in a short computational time. One example of heuristic that is frequently encountered in the lot-sizing literature is the so-called Relax-and-Fix (RF) procedure. In a nutshell, it is a constructive heuristic that builds a feasible solution using a decomposition strategy over the time horizon. It is often combined with a second step, called Fix-and-Optimize (FO), that aims at improving the quality of the initial solution using a local search procedure. Both approaches are quickly introduced and adapted to the CLSSD-PM in section 4.1.

It turns out that a simple adaptation of a RFFO heuristic does not yield satisfactory results in the allocated time on large instances. Since sequence-dependent setup times make the problem significantly harder to solve, we then take advantage of the data structure to overcome these limitations. Specifically, we introduce in section 4.2 a clustering approach that allows us to derive a variation of the original problem. Thus, it becomes easier to solve, through a significant reduction of the number of sequence

variables w_{mt}^{ij} and subtour elimination constraints. Combining this approach with the RFFO procedure, we are then able to obtain good solutions to this modified problem in a limited amount of time. In addition, the production plans obtained remain provably feasible for the original CLSSD-PM.

4.1. *A Relax-and-Fix and Fix-and-Optimize procedure*

We present two well-known decomposition-based heuristics: The Relax-and-Fix (RF) and the Fix-and-Optimize (FO). Both procedures rely on the resolution of multiple "easier" subproblems. The goal of each subproblem is to set the value of a small subset of integer variables, while the remaining ones are either relaxed or fixed.

4.1.1. Relax-and-Fix with period-based decomposition

The RF heuristic constructs a feasible solution iteratively by solving several subproblems with fewer binary variables. Several types of such decomposition exist in the literature, however we focus on the period-based decomposition which has been widely applied and is generally considered the most efficient method (see Helber and Sahling (2010) for a comparison with the product-based decomposition). The basic principle of the procedure consists in moving a decision window of δ periods through the planning horizon. The periods before and after the decision window are said to be *frozen* and *approximate*, respectively.

Specifically, during the first iteration the algorithm solves the MIP-AGG (or MIP-FL) problem considering binary variables w_{mt}^{ij} in the decision window ($t = 1, \ldots, \delta$) and their relaxed counterpart in the approximate periods, i.e. $0 \leq w_{mt}^{ij} \leq 1$ for all $t = \delta + 1, \ldots, T$. In the second iteration, the decision window is shifted to the right by $\delta - \sigma$ periods, where $\sigma < \delta$ is the number of overlapping periods in the decision window between two consecutive iterations. The setup variables from the previous approximated periods that enter the decision window are reverted to binary variables. Binary variables that leave the decision window, i.e. whose time index belongs to $\{1, \ldots, \delta - \sigma\}$, become frozen and keep their value from the previous resolution step. The algorithm repeats the procedure of shifting and fixing variables until the decision window reaches the end of the horizon. Figure 2 is inspired from Absi and Kedad-Sidhoum (2007) and represents the different time windows during two consecutive iterations of the heuristic.

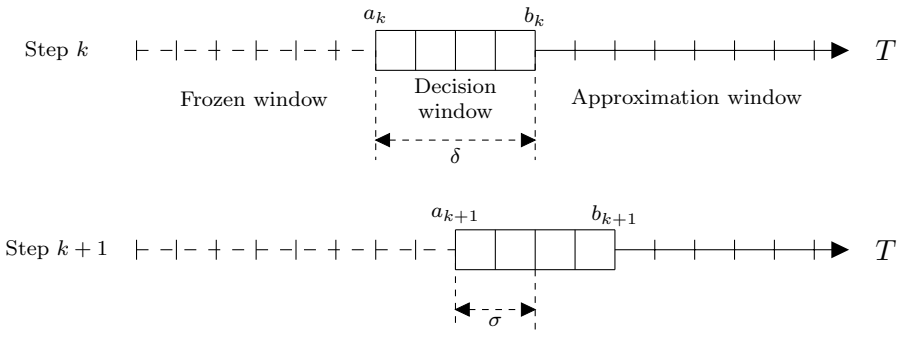


Figure 2. Horizon decomposition in the RF heuristic

We define $sol^k(x)$ as the value of the decision variable x in the solution obtained at iteration k of the algorithm and denote $a_k = 1 + (k-1)(\delta - \sigma)$ and $b_k = k\delta - (k-1)\sigma$

the first and the last period of the associated decision window, respectively. Let R_k be the corresponding relaxed subproblem, which is solved using either the aggregate or the facility location formulation introduced in Section 3. Problem R_k is similar to the original one, except for the variables definition constraints (9) that are replaced by the following ones :

$$w_{mt}^{ij} \in \{0,1\} \quad \forall i, \forall j, \forall m, \forall t \in [a_k, b_k] \tag{30}$$

$$w_{mt}^{ij} \in [0,1] \quad \forall i, \forall j, \forall m, \forall t \in [b_k + 1, T] \tag{31}$$

$$w_{mt}^{ij} = sol^{k-1}(w_{mt}^{ij}) \quad \forall i, \forall j, \forall m, \forall t \in [1, a_k - 1], \forall k \geq 2 \tag{32}$$

In order to control the total time necessary to obtain a solution with the RF procedure, we add two parameters that limit the computational time allocated to each subproblem R_k. Specifically, we first set a target gap γ for the solution of R_k, i.e. in each iteration k the solver returns the best solution found with a gap lower than or equal to γ. To avoid situations in which reaching γ induces a long computational time, we also set a time limit ρ for each iteration, i.e. the algorithm returns the best feasible solution obtained in ρ units of time, regardless of the gap achieved.

Note that the RF heuristic we describe in Algorithm 1 uses a forward period decomposition. A backward version, which starts at the end of the horizon and finishes at the first period, has also been tested in the literature (Toso, Morabito, and Clark (2009)). Since the two versions displayed equivalent performances during our preliminary tests, we focus on the forward version in our numerical experiments.

Algorithm 1 Relax-and-Fix heuristic

procedure RF($\delta, \sigma, \gamma, \rho$)
  $k \leftarrow 1, a_k \leftarrow 1, b_k \leftarrow \delta, sol^k \leftarrow \emptyset$
  while $b_k < T$ **do**
    $sol^k \leftarrow$ Solve R_k with maximum gap γ within ρ units of time
    $a_{k+1} \leftarrow b_k - \sigma + 1$, $b_{k+1} \leftarrow \min\{b_k + \delta - \sigma, T\}$, $k \leftarrow k + 1$
  end while
  Solve R_k
  Return sol^k
end procedure

4.1.2. Fix-and-Optimize

The RF approach is frequently combined with a local search procedure called Fix-and-Optimize to improve the feasible solution found. As for the RF heuristic, the procedure relies on a decomposition of the original problem into simpler subproblems, where the most classic decomposition is again based on periods. Starting from an initial reference solution, the idea is to "release" the subset of its binary variables whose time index belongs to a given decision window and reoptimize over them in order to improve the solution. If a better solution is found, it becomes the new reference and is used as such in the next iterations. During each iteration, binary variables that do not belong to the subset mentioned above are fixed either to their initial value or to a new value set in a reference solution from a previous iteration. At each iteration, only the binary variables (i.e setup variables) outside the decision window are fixed when others have to be reoptimized.

4.1.3. RFFO applied on the CLSSD-PM

The combination of RF and FO heuristics, denoted RFFO, is particularly efficient for complex lot-sizing problems and usually offers a good trade-off between execution time and solution quality. For instance, James and Almada-Lobo (2011) present these two approaches for the capacitated lot-sizing problem with sequence-dependent setups on parallel machines and obtain good results in short computational time. In both case, the size of the decision window greatly influences the computational time of the procedure. The different steps of the proposed RFFO procedure are decribed below :

Initialization Define the parameters $\delta, \sigma, \gamma, \rho$ for the RF and FO steps.
Step 1 Apply the Relax-and-Fix heuristic RF$(\delta, \sigma, \gamma, \rho)$ to obtain a feasible solution.
Step 2 Apply the Fix-and-Optimize heuristic on the current solution with the same parameters $\delta, \sigma, \gamma, \rho$.

An analysis of the impact of the different parameters (i.e the decision window size, the overlap between two consecutive decision windows, the limiting gap and time during each iteration) is presented in Section 5. In the remainder of the paper, we extend the notation introduced in the RF algorithm and denote RFFO$(\delta, \sigma, \gamma, \rho)$ the RFFO procedure using the parameters $\delta, \sigma, \gamma, \rho$ in Step 1 and Step 2.

4.2. *Clustering approach*

As mentioned in the previous section, SD setups are computationally challenging when one wishes to obtain a good solution in a reasonable time, even when they rely on fast heuristics. In fact, the previous results obtained in the simpler framework of a single-machine problem provide evidences that SD setups are much harder to tackle than the traditional item-dependent setup. In contrast with this setting, we call *sequence-independent* (SI) a setup time that only depends on the next item to produce on a given machine, i.e. where $\lambda_m^{ij} = \lambda_m^j$ is independent of i for all $(i, j, m) \in \mathcal{N}^2 \times \mathcal{M}$.

Our main idea is to find a way to transform an instance I of CLSSD-PM into a smaller instance $\tilde{I}$, for which some of the setups are considered sequence-independent. Since this operation may remove crucial information on the true setup times associated with a production sequence, we shall ensure that (i) there exists a feasible production sequence on each machine in each period that contains the same quantity of each item within the capacity available, and (ii) the SI setup times associated with a production plan for $\tilde{I}$ remain comparable to their SD counterpart in I.

We build our conversion procedure on a simple observation on the setup times characteristics that is verified in many real-life manufacturing problems encountered in food-industries. Indeed, the majority of switching times between items observed in practice are either "long" (40 minutes to 2 hours) or "short" (1 to 10 minutes) and satisfy the triangle inequality. For instance, let's consider the case of a manufacturer that produces three types of cookies. Type 1 cookies contain allergen ingredients while types 2 and 3 are both free of allergens. The cleaning time between type 1 and any of the two others will likely be significantly longer than the one between types 2 and 3, and extend the setup times accordingly. Similar asymmetries may also occur from differences in oven temperatures, organic/non-organic product, etc.

In an attempt to reduce the global setup times and maximize resources utilization, practitioners often adopt an intuitive scheduling policy and try to regroup items with similar ingredients or recipe characteristics in the production sequences, with the objective to limit the number of long setup times in their production sequences. The

main driver of the total setup time then becomes the arrangement of these group of items, rather than the sequence of items within a particular group. Our approach takes advantage of this specificity and use the data available on the setup times to build logical families of items. This allows us to define approximate instances for which we only consider sequence-dependent switching times between families, while intra-family item setup times are approximated by virtual, sequence-independent ones. In Robinson and Lawrence (2004), a problem including major family setups and minor setups are presented for the coordinated replenishment lot-sizing problem but the authors do not consider the sequence-dependent features. This approach reduces significantly the number of sequence-based decision variables, therefore lowering the size of the problem.

4.2.1. A clustering approach

Our procedure consists in partitioning the items into clusters that gather items together into families, such that the setup time between any two items in the same cluster is negligible compared to the setup time between two items that belong to different clusters. To this end, we rely on a partitioning algorithm called PAM (Partitioning Around Medoids) sometimes referred to as k-medoids, whose only inputs are an integer k and the setup times between all pairs of items. Hence PAM only requires a matrix of distances for each pair of points, a desirable feature compared to alternative procedures such as the well-known k-means that requires to define the coordinates of every point in a normed space. Note that the original PAM algorithm considers symmetric distances to determine which medoid is the closest to a given point. To meet this requirement while ensuring the robustness of our approach, we define a virtual setup time $\tilde{\lambda}_m^{ij} = \max\{\lambda_m^{ij}, \lambda_m^{ji}\}$ for all $i, j \in \mathcal{N}$ and $m \in \mathcal{M}$. Thus two items with asymmetric changeover times (caused by organic components for instance), are likely to end up in two different clusters if the maximum value of their changeover times is large.

Although PAM is a well-known algorithm in the Machine Learning community, we quickly describe it below to make the paper self-contained. In a nutshell, the procedure builds clusters in an iterative fashion around specific points (items), considered as the medoid (center) of their respective cluster. At any given stage of the algorithm, the quality of the current partition on a given machine $m \in \mathcal{M}$ is evaluated using a cost function E_m that is defined from the virtual setup times as follows:

$$E_m = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{C}_i} \tilde{\lambda}_m^{ij} \tag{33}$$

, where $\mathcal{I} \subseteq \mathcal{N}$ is the set of medoids in the current solution and for all $i \in \mathcal{I}$, $\mathcal{C}_i$ is the set of items that belong to the cluster associated with i. The procedure starts with an initial partition based on k randomly selected medoids as the reference and proceeds by performing successive swaps between one of the current medoid and other (non-medoid) items. It then assigns every point to its closest medoid and takes this new configuration as the new reference if it improves the quality of the clusters obtained (based on the value of E_m). The procedure repeats the previous steps until no further improvement is observed. Algorithm 2 presents the PAM algorithm.

As mentioned earlier, one of the inputs required by PAM is the number k of clusters to define. It thus remains to estimate a value of k that ensures a relevant partition of items. To this end, we use an indicator based on the so-called silhouette score for

Algorithm 2 PAM

procedure PAM(k, $\mathcal{N}$, m)
  actualCost $= +\infty$, $\mathcal{I} \leftarrow \emptyset$
  Choose randomly k items $\in \mathcal{N}$ to be the first medoids and update $\mathcal{I}$
  Associate each item $i \in \mathcal{N} \setminus \mathcal{I}$ to the closest medoids
  actualCost = Compute E_m
  while actualCost decrease **do**
    for i in $\mathcal{I}$ **do**
      for o in $\mathcal{N} \setminus \mathcal{I}$ **do**
        Compute E_m for the set of medoids $\mathcal{I}' = \mathcal{I} \cup \{o\} \setminus \{i\}$
        if $E_m <$ actualCost **then**
          actualCost $\leftarrow E_m$
          $\mathcal{I} \leftarrow \mathcal{I}'$
          Go to 6
        end if
      end for
    end for
  end while
  Return $\mathcal{I}$
end procedure

every item, which measures its matching quality with the other items that belong to the same cluster. Formally, let $m \in \mathcal{M}$ and $i \in \mathcal{I}$: The score obtained by an item $j \in \mathcal{C}_i$ relies on a combination of the average distance between j and other items in $\mathcal{C}_i$, versus the average distance between j and the items in the cluster $\mathcal{C}_{i'}$ closest to j for $i' \neq i$. Let us define $a_m(j)$ as the average setup time from item j to other items in cluster $\mathcal{C}_i$ on machine m:

$$a_m(j) = \frac{1}{|\mathcal{C}_i| - 1} \sum_{j' \in \mathcal{C}_i, j' \neq j} \tilde{\lambda}_m^{jj'} \tag{34}$$

and $b_m(i)$ as the average setup time from the closest cluster of j on machine m:

$$b_m(j) = \min_{i' \neq i} \frac{1}{|\mathcal{C}_{i'}|} \sum_{j' \in \mathcal{C}_{i'}} \tilde{\lambda}_m^{jj'} \tag{35}$$

The silhouette score of item j, noted $s_m(j)$ is then obtained from (34) and (35):

$$s_m(j) = \frac{b_m(j) - a_m(j)}{\max\{a_m(j), b_m(j)\}} \tag{36}$$

Note that the closest to 1 the silhouette score is, the better the item is positioned in the cluster. One often considers having a convenient clustering for average silhouette higher than 0.75. To obtain more details on the meaning of the silhouette score in a clustering approach, see Laan, Pollard, and Bryan (2003) and Campello and Hruschka (2006).

In order to select the best number of clusters k_m^* for each machine m, we loop over the possible values of k and apply the PAM algorithm at each iteration. For a given k

and a given machine $m \in \mathcal{M}$, we consider the output of the procedure and compute the *average silhouette criterion* as the mean of the silhouette score over all items:

$$S_m(k) = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} s_m(i) \tag{37}$$

The higher this average score is, the better is the clustering quality for the machine we consider. We select the clustering with the greatest average silhouette, which is considered as the best possible on the machine for the given instance. In Section 5, we show that $S_m(k)$ is a good *a priori* indicator to obtain the best trade-off between computational time and solution quality. For a quick description of the procedure, we present the pseudo-code of the clustering pre-processing approach in Algorithm 3. Note that for a CLSSD-PM instance, we run this algorithm $|\mathcal{M}|$ times since we consider machine-dependent setups.

Note that we use PAM as a pre-processing procedure before the definition of the MIP model. Its runtime is negligible (less than 15 seconds for the largest instances) compared to the one necessary to the solver to obtain a solution to the CLSSD-PM.

Algorithm 3 Clustering

```
1: procedure CLUSTERING(maxClusters, 𝒩, m)
2:     actualClustering ← ∅, bestClustering ← ∅, actualScore ← 0, bestScore ← 0,
3:     for k = 2, ..., maxClusters do
4:         actualClustering ← Run PAM(k, 𝒩, m)
5:         actualScore ← Compute AvgSilhouette(actualClustering)
6:         if actualScore > bestScore then
7:             bestScore ← actualScore
8:             bestClustering ← actualClustering
9:         end if
10:    end for
11:    Return bestClustering
12: end procedure
```

4.2.2. Cluster-based mathematical formulation

We now define a virtual setup time $v_m^j = \max_{k \in \mathcal{C}_i}\{\lambda_m^{kj}\}$ for each item $j \in \mathcal{C}_i$, which only depends on the end point of the corresponding arc in the directed path representing the production sequence. For each pair of medoids (i, i') and for each machine we also define a new sequence-dependent setup time $f_m^{ii'} = \max_{j \in \mathcal{C}_i, j' \in \mathcal{C}_{i'}}\{\lambda_m^{jj'} - v_m^{j'}\}$ incurred when an item of $\mathcal{C}_{i'}$ follows an item of $\mathcal{C}_i$ in the production sequence. Note that this setup time excludes the end item individual setup time defined above, since the latter is added to the total setup time if the item is indeed included in the production sequence.

With these new virtual setup times, we are now able to only consider sequence-dependent setup times when the production switches from one cluster to another. In particular we no longer take into account the production sub-sequences within a cluster and therefore replace the transition variables w_{mt}^{ij} with traditional setup binary variables y_{mt}^j. In our modified model, the switching decision between two clusters is represented by new binary variables $z_{mt}^{ii'}$, equals to 1 if cluster i' is setup just after cluster i on machine m in period t and 0 otherwise. On the other hand, whenever an item

is produced on machine m in a given time-period, we only account for its item-specific setup time v_m^i in the total active time of the machine. Specifically, constraints (3) to (8) can be directly adapted to take into account the clusters instead of the items in the sequences of production. Constraints (12) are replaced by :

$$\tilde{U}_{mt} = \sum_{j \in \mathcal{N}} (\tau_m^j x_{mt}^j + v_m^j y_{mt}^j) + \sum_{(i,i') \in \mathcal{I}^2} f_m^{ii'} z_{mt}^{ii'} \leq C_{mt} + O_{mt} \quad \forall m = 1, \ldots, M, \forall t = 1, \ldots, T \tag{38}$$

The constraint (25) for the facility location formulation is also replaced in an equivalent fashion. We add (39) to force a cluster setup if we want to produce an item in the cluster :

$$\sum_{i \in \mathcal{I}} z_{mt}^{ii'} \geq y_{mt}^j \quad \forall i' \in \mathcal{I}, \forall j \in \mathcal{C}_{i'}, \forall m = 1, \ldots, M, \forall t = 1, \ldots, T \tag{39}$$

We denote MIP-AGG-C and MIP-FL-C the modified versions of problems MIP-AGG and MIP-FL, respectively. It remains to prove that one can easily build a feasible solution to MIP-AGG (resp. MIP-FL) from a feasible solution to MIP-AGG-C (resp. MIP-FL-C) without any overcost.

We prove this property for MIP-AGG, but a similar analysis can be conducted to obtain the same results for MIP-FL. Let $(\mathbf{x}, \mathbf{z}, \mathbf{y}, \mathbf{O})$ be a solution to MIP-AGG-C. For all $j \in \mathcal{N}$, define $\mu(j)$ as the medoid of the cluster of item j, i.e. $j \in \mathcal{C}_{\mu(j)}$, obtained after applying the PAM clustering algorithm. We propose the following simple procedure to create a solution to MIP-AGG: For all $m \in \mathcal{M}$, $t \in \mathcal{T}$, we define the following transition variables:

$$w_{mt}^{0j} = \begin{cases} 1 & \text{if } j = \min\{j' \in \mathcal{N} : z_{mt}^{0\mu(j')} = 1\} \\ 0 & \text{otherwise} \end{cases} \tag{40}$$

$$w_{mt}^{jj'} = \begin{cases} 1 & \text{if } j' = \min\{j'' \in \mathcal{C}_{\mu(j)}, j'' > j : y_{mt}^{j''} = 1\} \\ & \text{or } z_{mt}^{\mu(j)\mu(j')} = 1 \text{ and } j' = \min \mathcal{C}_{\mu(j')} \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

The following proposition states that the above transformation yields a feasible solution to MIP-AGG:

Proposition 4.1. Let $\mathbf{s}_C = (\mathbf{x}, \mathbf{z}, \mathbf{y}, \mathbf{O})$ a solution to MIP-AGG-C. Then $\mathbf{s} = (\mathbf{x}, \mathbf{w}, \mathbf{O})$ with $\mathbf{w} = \left(w_{mt}^{ij}\right)_{i,j \in \mathcal{N}, m \in \mathcal{M}, t \in \mathcal{T}}$ defined from equations (40) and (41) is a feasible solution to MIP-AGG that incurs at most the total cost of $\mathbf{s}_C$.

Proof. Let $m \in \mathcal{M}$ and $t \in \mathcal{T}$ and compute the total active time on machine m in

period t in solution $\mathbf{s}$

$$
\begin{aligned}
U_{mt} &= \sum_{j\in\mathcal{N}} \left(\tau_m^j x_{mt}^j + \sum_{i\in\mathcal{N}} \lambda_m^{ij} w_{mt}^{ij} \right) \\
&= \sum_{j\in\mathcal{N}} \left(\tau_m^j x_{mt}^j + \sum_{i\in\mathcal{N}} (\lambda_m^{ij} - v_{mt}^j + v_{mt}^j) w_{mt}^{ij} \right) \\
&\leq \sum_{j\in\mathcal{N}} \left(\tau_m^j x_{mt}^j + v_{mt}^j \sum_{i\in\mathcal{N}} w_{mt}^{ij} + \sum_{i\in\mathcal{N}} \max_{i'\in\mathcal{C}_{\mu(i)}} \{\lambda_m^{i'j} - v_{mt}^j\} w_{mt}^{ij} \right) \qquad (42) \\
&= \sum_{j\in\mathcal{N}} \left(\tau_m^j x_{mt}^j + v_{mt}^j y_{mt}^j + \sum_{i\notin\mathcal{C}_{\mu(j)}} \max_{i'\in\mathcal{C}_{\mu(i)}} \{\lambda_m^{i'j} - v_{mt}^j\} w_{mt}^{ij} \right) \qquad (43) \\
&= \sum_{j\in\mathcal{N}} \left(\tau_m^j x_{mt}^j + v_{mt}^j y_{mt}^j \right) + \sum_{j\in\mathcal{N}} \sum_{i\notin\mathcal{C}_{\mu(j)}} \max_{i'\in\mathcal{C}_{\mu(i)}} \{\lambda_m^{i'j} - v_{mt}^j\} z_{mt}^{\mu(i)\mu(j)} \\
&= \sum_{j\in\mathcal{N}} (\tau_m^j x_{mt}^j + v_m^j y_{mt}^j) + \sum_{(i,i')\in\mathcal{I}^2} f_m^{ii'} z_{mt}^{ii'} \\
&= \tilde{U}_{mt} \leq C_{mt} + O_{mt}
\end{aligned}
$$

Inequality (42) comes from the fact that $\max_{i'\in\mathcal{C}_{\mu(j)}}\{\lambda_m^{i'j} - v_{mt}^j\} = 0$, while (43) follows from the definition (41) of w_{mt}^{ij}.

To conclude the proof, notice that all variables with nonzero coefficients in the objective function are the same in $\mathbf{s}$ and $\mathbf{s}_C$, except $U_{mt} \leq \tilde{U}_{mt}$ whose coefficient is positive. Hence the total cost incurred by $\mathbf{s}$ is at most equal to the total cost incurred by $\mathbf{s}_C$. □

In general, the partition of items into clusters reduces significantly the number of decision variables related to the sequence of production. In the cluster version of CLSSD-PM, the latter becomes quadratic in the number of clusters, as opposed to quadratic in the number of items in the original version.

Our definition of the virtual setup times ensures that any production sequence of items intra *and* inter-cluster will be lower than or equal to $\tilde{U}_{mt}$, as defined in (38), hence it is always possible to construct a feasible solution with at most a production time $\tilde{U}_{mt}$ for all $m \in \mathcal{M}$ and $t \in \mathcal{T}$. In particular, any algorithm that reconstructs intra-cluster production sequences (random, closest neighbor, etc.) from a feasible solution to MIP-AGG-C and then concatenates them according to the sequence of clusters corresponding to variables $z_{mt}^{ii'}$ is guaranteed to be feasible for MIP-AGG. In our computational experiments, we use a greedy algorithm to reconstruct the optimal production sequences. Finally, note that in addition to the flexibility that the method offers to the scheduler, one may also take advantage of such reductions in production time to decrease production costs or even to serve demands that are considered lost in MIP-AGG-C.

5. Computational experiments

The data that our industrial partner could gather from its clients was neither sufficiently complete nor precise enough to perform a thorough numerical study. We therefore chose to generate a new set of instances based on the information provided and use them to compare the different methods introduced above. Data sets are inspired by different real cases from food-industry producers and are built to cover a great variety of situations that one can expect to encounter in practice. One common characteristic of these instances lies in the structure of the setup times, which permits to apply the clustering approach described in the previous section. Although this assumption may seem restrictive, it reflects the distinctive feature of production processes in the food-industry.

The algorithms are coded in Java and we use IBM Ilog CPLEX 12.10.0 version as the MIP solver The experiments are performed using four threads on an Intel Xeon X5650 @ 2.57 GHz. For CPLEX and CPLEX-C, we set a computational time limit of 900 seconds. For each method tested, we evaluate the solution quality based on the deviation of its objective value from the lower bound LB obtained after 4 hours with CPLEX using the MIP-FL formulation, known to yield better lower bound. If UB corresponds to the best feasible solution found by the method tested, the gap is then defined as follows:

$$\text{Gap} = 100 \cdot \frac{UB - LB}{LB} \tag{44}$$

We use the following abbreviations to identify the algorithms used in our test :

- CPLEX : branch-and-cut by CPLEX on MIP-AGG
- CPLEX-C : branch-and-cut by CPLEX on MIP-AGG-C
- RFFO : Relax-and-Fix & Fix-and-Optimize heuristic on MIP-AGG
- RFFO-C : Relax-and-Fix & Fix-and-Optimize heuristic on MIP-AGG-C

The RFFO results presented in this section use the best settings obtained in section 5.2, i.e. the ones reaching the best trade-off between solution quality and computational time. After preliminary tests, we also observed that the solution methods that rely on the MIP-AGG formulation obtain better feasible solutions compared to their MIP-FL counterpart. Hence, we focus on the former and only use the latter to compute the best possible lower bound.

5.1. *Instance generation*

Problem size. Recall that our main goal is to obtain good solutions in a short computational time on large instances. The time limit is an important aspect for our industrial partner and is loosely set to a few minutes. We agreed afterwards to set it to 180 seconds, with a tolerance to up to 300 seconds for some instances (namely with more than 100 different products). In order to determine the influence of problem size (based on the number of items, machines and periods) on both the quality of the solution and the computational burden, we generated several instances for different combinations of values for N, M and T. We then partitioned the instances obtained into three classes (small, medium and large), presented in Table 3. In order to test the performances of the clustering approach in different settings, we use two different approaches that generate symmetric and asymmetric sequence dependent setup times,

respectively. We create six instances for every combination of N, M and T, giving us a total of 336 instances.

Table 3. Class instance size

Class	N	M	T	Max. time (s)
Small	{20,30,40}	$\{1,2\}$	$\{15,30\}$	180
Medium	{50}			
	{75,100}	{2,4}		300
Large	{125}	{4,6}		

Definition of problem parameters. We derive our instances from the information provided by our industrial partner, with the objective to build data sets as close as possible to real-life cases. The parameters are defined as follows:

Time Capacity. The production time capacity on each machine, in each period is either 8, 16 or 24 hours. The maximum overtime is equal to 2% of the total capacity but is not considered in the latter case.

Processing time for multiple machines. For instances with 2 machines or more, we perform for each item i an independent Bernouilli trial to decide whether it can be produced on several machines or not, with a success probability drawn uniformly in $[0.25, 0.5]$. In case of success, we draw uniformly between 2 and M the number M_i of machines compatible with i and randomly select their indices as a subset of $\{1, \ldots, M\}$ of size M_i. Whenever a machine can produce a given item, the per-unit processing time is drawn uniformly in $[0.05, 0.5]$ minutes.

Demand. We first determine how the total cumulative demand is allocated across the different items. Specifically for each item i, we draw a random number from a uniform distribution $U(0, 1)$ and normalize the values obtained so that they sum up to 1. We then define the demands in each period t in an iterative fashion, assigning each unit to a specific item according to the probabilities previously defined. The procedure adds units of demand one after another as long as $\sum_i \min_m\{\tau_{im}\} d_t^i / \mathit{Util} < \sum_m C_{mt}$, where $\mathit{Util} \in \{0.6, 0.8\}$ represents the capacity utilization of the machines (decided beforehand as an input to the instance generator). Finally, we consider that 25% of the items present a periodic demand pattern, in which periods with null demand are regularly spaced in the planning horizon.

Setup times. As mentioned earlier, we define two types of setup times, expressed in minutes. In what follows, "short" setup times are drawn uniformly in the interval $[1, 10]$, while "long" ones are drawn uniformly in the interval $[40, 120]$. To ensure they satisfy the triangle inequality, we initialize their values to zero and instantiate them sequentially: In each step of the procedure, we draw λ_m^{ij} in the interval $[\max\{l_{ij}, |\lambda_m^{kj} - \lambda_m^{ik}|\}, \min\{u_{ij}, \lambda_m^{ik} + \lambda_m^{kj}\}]$ for any triplet $i, j, k \in \mathcal{N}$, where l_{ij} and u_{ij} represent the minimum and the maximum of the corresponding (i.e. short or long) domain. We distinguish two classes of instances depending on how the setups are generated:

- Instances in the first set are strongly related to real applications. In the food industry, an item is often characterized by several features (for instance organic, specific oven temperature or food allergens) that influence the setup times. In practice, it is then the change in features between an item and its successor in the production sequence that effectively defines the length of this time interval. Based on this observation, we define 4 features for each item. Each feature has either 2 (e.g. organic and non organic) or 3 (e.g. oven temperatures) possible states. We then draw the setup times associated to

each features randomly in the "short" or the "long" interval, depending on the nature of the feature considered. For the same feature, the switching time between two states may be asymmetric: A typical example arise when we set up a machine for allergen-free items after a production that contains allergen, a longer operation than the one performed for the opposite transition. As the setup operations corresponding to each specification are usually performed in parallel, we define the changeover time between two items as the maximum duration among all the switching times necessary for their respective features states.

– We build the instances of the second set with the following procedure: (1) We (randomly) choose a number of subsets, (2) We randomly assign each item to a subset and (3) We assign short setup times between all pair of items within the same subset and long setup times when they belong to different ones, ensuring in the process that the triangle inequality is always satisfied. This procedure guarantees that we obtain the clustering structure that is encountered in the food industry and for which our clustering approach behaves well.

Safety stock. The safety stock of an item i in period t corresponds to the sum of demands for i over future periods $t+1, \ldots, t+\Delta_{it}$. The number Δ_{it} of future periods considered is often called safety cover and is chosen according to a discrete uniform distribution $U(1,3)$. It can evolve over time with a low probability, but always includes the cover of the previous period (i.e $\Delta_{i,t+1} \geq \Delta_{it} - 1$ for all i, t).

Minimum production quantity. This quantity is generated from a uniform distribution $U(1,3)$.

Table 4 describes the generation of cost parameters. We consider an overcost equal to 50% of the regular line use cost. The lost sales cost for a given demand is built from the greatest possible cumulated production cost and holding cost incurred to produce the corresponding units. This ensures that serving a demand is always profitable whenever it is possible.

Table 4. Cost parameters

Parameter	Settings
c_{mt}	$U(500, 1000)$
$\bar{c}_{mt}$	$0.5c_{mt}$
h_u^{i+}	$U(1,3)$
h_u^{i-}	$U(15,30)$
p_{mt}^i	$U(0.2, 0.5)$
l_t^i	$\max_{s \leq t,m,j}\{p_{ms}^i + (c_{mt} + \bar{c}_{ms})(\tau_m^i + \frac{\gamma_m^{ji}}{q_{\min}^i}) + \sum_{u=s}^{\theta_t^i - 1} h_u^{i+}\}$

5.2. *RFFO parameters analysis*

The performances of the RFFO procedure heavily depends on the values δ, σ, γ and ρ. Hence it is crucial to determine the combination of parameters that provides the best expected trade-off between the cost of the solutions obtained and the average computational time for each class of instances. After an empirical preliminary analysis, we decided to focus on a limited number of settings. We set the number of overlapping periods σ to 1 since increasing its value does not improve significantly the solutions but severely deteriorates the computational time for medium and large instances. Even for small instances, we did not observe a substantial gain in performances for $\sigma > 1$. We tested different combinations of decision window length $\delta \in \{2, 3\}$, gap tolerance (given

by CPLEX) $\gamma \in \{1, 3, 5, 8, 10\}\%$ and time limit per iteration $\rho \in \{30, 60, 90, 120\}$ seconds.

In order to decide which stopping criteria γ and ρ perform best, we analyzed their impact on the solutions. Note that this analysis was made on RFFO-C as RFFO often failed to find a feasible solution during the RF part on many instances. The results for medium size instances are shown in Appendix A. As expected, the gap of the final solution increases with γ, but this influence is mild when $\gamma \leq 5\%$ (Figure A1) On the other hand the influence of γ on the computational time is more pronounced, for instance setting $\gamma = 3\%$ reduces significantly the computational time compared to $\gamma = 1\%$. The former value thus provides the best compromise between the final gap and computational time. A similar analysis is conducted for different values of ρ. On the other hand, we observe in Figure A3 that the time limit is never exceeded for instances with $N = 50$. For other instances, the results suggest that ρ has a low impact on both the total computational time and the final gap. Hence, according to the results, $\rho = 60$ seems to be consistent as the time limit is seldom the limiting parameter when γ is adjusted appropriately.

The tables in Appendix B presents the results using RFFO on each class of instances. For each RFFO settings presented, we let $\sigma = 1$ and $\rho = 60$. We observe that $\delta = 3$ leads to a better gap on small and medium instances for all the γ values we tested.

- For $N \leq 50$, $\gamma = 1\%$ appears to produce the best solutions under 180 seconds in general. Hence we decided to apply RFFO-C$(3, 1, 1, 60)$ on instances with $N \leq 50$.
- For larger instances, $\gamma = 3\%$ turns out to be a good choice to configure RFFO-C, except when $N = 100$ and $T = 30$, where $\gamma = 5\%$ gives a better trade-off between performances and computational time.
- The effect of a bigger decision window is more visible on the largest instances ($N = 125$), with a significant deterioration of the gaps obtained on instances with $(N = 125, M = 6, T = 30)$. In fact when $N = 125$ and $T = 15$, setting $\delta = 2$ appears to be a more robust choice in general. We thus decided to retain RFFO-C$(2, 1, 5, 60)$ for $(N = 125, T = 15)$ and RFFO-C$(3, 1, 8, 60)$ for $(N = 125, T = 30)$ as the preferred configuration for the largest instances.

A summary of the RFFO configurations for the different types of instance is presented in Table 5.

Table 5. RFFO configuration

N	20		30		40		50		75		100		125	
T	15	30	15	30	15	30	15	30	15	30	15	30	15	30
γ (%)	1								3		5		8	
δ	3										2		3	

5.3. *Clustering approach results*

To investigate the benefits of our clustering approach, we performed computational experiments using CPLEX directly and applying the RFFO procedure on both MIP-AGG and MIP-AGG-C. The results are presented in Table 6 and 7. As expected, the direct use of a commercial solver on the MIP formulation turns out to be inadequate, even for some instances with 30 items. We allow CPLEX a maximum computational time of 900 seconds, which already exceeds the industrial requirements of 3 to 5 min-

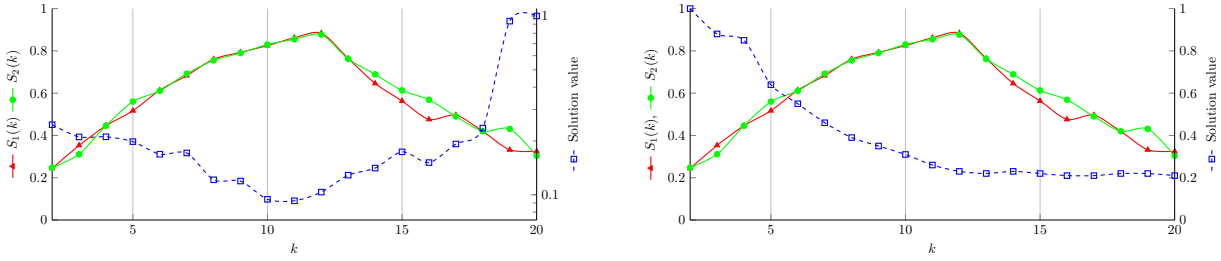
utes. While the solver is able to derive an optimal solution on some of the instances, the quality of the production plans obtained with this approach is not satisfying on average. Based on our observation, each acceptable solution from CPLEX was found towards the end of that time limit, which increases the variability of the solutions and makes it incompatible with an industrial approach.

In contrast, the clustering pre-processing reduces the size of the problem and allows to find significantly better solutions in most scenarios, with a faster convergence rate. The idea is that the loss in solution quality due to the approximation is compensated by a faster resolution. The results obtained suggest that it is the case for instances with $N \geq 40$, for which CPLEX-C reaches better solutions than its sequence-dependent counterpart. We observe that this tendency becomes more and more pronounced as N increases. Finally, it appears that the length of the planning horizon T affects significantly the solution quality for both approaches.

We also test whether the silhouette score introduced in section 4.2 is a relevant criterion to drive the number of families considered. That is, we evaluate the solutions obtained by MIP-AGG-C for several number of clusters k and compare their quality with the corresponding average silhouette criterion $S_m(k)$ as defined in (37). The general conclusion is that the cost of the solution obtained decreases as this number grows to its "best" value $k^* = \arg\max\{S_m(k)\}$. When the number of clusters grows beyond k^*, the solutions obtained either deteriorate significantly or show mild improvement. Indeed as the number of families increases, we introduce more sequence-dependent setup variables in the formulation MIP-AGG-C, which makes the problem closer to the original one but harder to solve. In our case, the silhouette criterion thus appears to be a suitable and easy-to-compute indicator of when this trade-off is favourable. Figure 3 illustrates this tendency on the worst case we observed across the instances generated for $M = 2$ and $N = 100$ for different number of clusters. The red and green curves show the average silhouette score $S_m(k)$ obtained on machine $m \in \{1, 2\}$ for different number of clusters k. The blue curve provides an indicator of the solution quality, expressed as the ratio with the worst solution obtained across all the possible number of families considered. The quality of the solution clearly improves as the number of clusters increases towards k^*. In the case of the RFFO resolution, we see that slightly better solutions are found for a larger number of clusters, but the increase in computational time is significant. On the contrary for a straightforward CPLEX resolution in 900 seconds, we observe that beyond a certain point, the solution quality deteriorates as we consider more clusters. We conclude that this criterion is a viable candidate to choose a good number of clusters a priori in order to obtain a good solution.

The experiments for the RFFO procedures are performed with the values δ, σ, γ and ρ adjusted for each class of instances according to the analysis in the previous section. In Table 6, we clearly observe a reduction of the average gap compared to a straightforward MIP resolution. On small instances, the gap obtained by RFFO combined with the clustering approach remains low compared to the others methods. In general, we observe that combining these two approaches significantly reduces the computational time while providing acceptable to good solutions on all the instances.

Despite performing better than CPLEX, the RFFO procedure directly applied to the CLSSD-PM problem generally leads to insufficient results or time overrun even on instances with $N = 40$. For larger instances, it often comes from incomplete iterations during the RF part of the heuristic, which cannot find a solution within the allocated time. A solution could be to increase the time limit ρ allowed for each iteration but this comes at the expense of a longer total computational time, which



(a) CPLEX-C (b) RFFO-C

Figure 3. Effect of the number of cluster on the solution quality and the silhouette score

then frequently exceeds the acceptable threshold set by the practitioners. In contrast both the computational time and the average gap obtained remain acceptable when we combine the RFFO procedure with the clustering approach. In Table 7 we compare the results obtained with CPLEX and RFFO on the modified problems obtained after the clustering pre-processing is performed. The clustering pre-processing defines a number of clusters varying between 5 and 20. The results on medium size instances, shown in Table 7, indicate that RFFO-C reaches better average gaps while maintaining the computational time well below 3 minutes on average. However the minimum gap (Gap_{min}) and maximum gap (Gap_{max}) obtained show that there is a great variation in the solution qualities even within a single class of instances. Regardless, RFFO-C has the best worst case performance among the two methods, with a maximum gap of 204.19% compared to the one of CPLEX-C which exceed 7000%. In addition, we observed that the average computational time for RFFO-C is sensitive to the different parameters and size of the problem. Finally, for the 125-6-30 problem size instances, we exceed the computational time objective, which calls for additional refinements of the methods to obtain a quicker convergence.

Table 6. Summary of the results obtained with each method on small instances

Problem Size	Avg. Gap (%)				Sol. Time (s)			
N-M-T	CPLEX	CPLEX-C	RFFO	RFFO-C	CPLEX	CPLEX-C	RFFO	RFFO-C
20-1-15	0.76	11.5	9.39	12.26	477.83	214.33	26.92	5.75
20-1-30	1.58	6.19	13.52	7.46	720.67	448.17	44.25	16.58
20-2-15	0.96	5.61	6.64	5.68	515.17	339.67	15.83	8.42
20-2-30	2.61	10.08	15.77	11.17	798.58	661.08	47.75	22.92
30-1-15	2.09	4.8	9.89	6.57	795.83	286.0	79.08	9.17
30-1-30	230.53	14.54	19.99	16.26	843.0	516.67	186.83	20.83
30-2-15	5.34	6.37	11.33	7.89	788.0	558.25	107.0	41.25
30-2-30	14.36	7.01	18.1	7.36	900.0	692.42	205.5	57.42
40-1-15	302.78	13.53	71.6	14.85	850.75	485.42	259.58	28.5
40-1-30	1984.33	102.33	65.24	23.27	900.0	756.17	516.33	108.75
40-2-15	27.39	7.75	18.01	9.35	812.33	618.33	203.92	37.0
40-2-30	1160.57	22.27	47.71	10.82	900.0	760.33	440.17	140.67

6. Concluding remarks and perspectives

We introduce two mixed-integer programming formulations for a production problem on multiple non-identical machines encountered in the food industry, in which setup

Table 7. Summary of the results obtained with CPLEX-C and RFFO-C on medium and large instances

Problem Size	CPLEX-C				RFFO-C			
N-M-T	Gap_{min}(%)	Gap_{max}(%)	Avg. Gap (%)	Sol. Time (s)	Gap_{min}(%)	Gap_{max}(%)	Avg. Gap (%)	Sol. Time (s)
50-1-15	1.2	59.26	23.2	511.92	1.85	56.62	24.7	48.5
50-1-30	0.16	186.68	51.19	702.25	0.35	56.39	23.22	87.0
50-2-15	0.17	33.16	9.62	568.67	1.32	33.72	9.56	85.08
50-2-30	2.14	95.63	19.39	900.0	3.98	35.35	12.42	122.0
75-2-15	0.45	121.87	41.83	785.17	2.22	119.97	40.03	49.75
75-2-30	6.2	805.77	163.09	900.0	8.73	161.9	48.29	106.58
75-4-15	0.66	149.64	42.07	900.0	3.12	73.5	27.69	146.33
75-4-30	0.79	3674.7	510.84	900.0	3.38	76.3	26.26	163.42
100-2-15	8.05	1051.87	211.85	848.92	9.56	169.51	69.92	140.75
100-2-30	5.57	4953.14	904.22	900.0	10.06	190.08	69.75	175.5
100-4-15	4.13	503.37	101.22	900.0	5.8	83.02	36.37	168.58
100-4-30	3.28	3954.51	713.25	900.0	8.9	143.64	54.52	166.33
125-4-15	2.24	3404.23	405.53	900.0	9.23	204.19	70.73	142.67
125-4-30	10.91	6488.41	1764.62	900.0	24.43	165.75	72.59	175.58
125-6-15	6.43	5696.75	954.55	900.0	11.28	131.28	57.09	171.92
125-6-30	6.25	7973.06	3863.41	900.0	19.25	164.1	78.78	351.58

times are sequence-dependent. Since the number of setup variables strongly influences the computational effort necessary to solve such problems, we take advantage of the characteristics of those changeover times to apply a clustering algorithm and derive approximate instances of reduced size. The loss due to the approximation is regained thanks to the effectiveness of the method on large industrial cases. When we combine this approach with an adaptation of the classical Relax-and-Fix and Fix-and-Optimize heuristics, we are able to derive acceptable solutions in a very short computational time relative to the problem complexity. Our experiments allow us to test several combinations of parameters and determine which configuration is susceptible to provide the best results for different classes of instances. In addition, we provide evidences that the silhouette score is a good indicator to define an appropriate number of clusters with respect to the final solution. Computational experiments confirm that our procedure performs well on medium to large industrial instances, for which we were able to obtain feasible production plans in less than 3 and 5 minutes, respectively. Our approach remains useful on well structured instances in which setup times are not uniformly distributed. In the general case however, it would likely be irrelevant since we observe that a poor silhouette score results in a poor solution.

The approximate version of the problem obtained after our pre-processing clustering algorithm remains quite rich and includes many industrial constraints. In this study, we mainly focus on the contribution of the clustering on the resolution time and quality, but we could refine our conclusions by examining in details the different influencing factors. In particular, investigating the impact of the setup times or the parameter *Util* on the solutions obtained may help to better explain the variability observed on the gaps for similar problem sizes. From an academic standpoint, evaluating the impact of each type of constraint such as the inclusion of safety stocks or overtimes may help to identify which subproblems benefit the most from the clustering transformation. One could also simplify even further the problem by defining sequence-independent setup times between clusters of items. However, a direct application of our method may loose too much accuracy since inter-cluster changeover times contribute a lot more than their intra-cluster counterpart to the total machine usage time. The virtual setup times that are currently defined as the maximum over all the possible changeover times towards a given item are likely to be too far from the true value achieved. Hence, one

should seek a different upper bound that retains more information on the production sequence to refine the approximation.

Finally, it is clear that all the solution methods presented in this paper are likely to take advantage of a strengthened MIP formulation. First, deriving a better lower bound would enable us to draw sharper conclusions on the quality of the solution obtained, especially on large instances. In addition, tightening the formulation is likely to speed up each iteration of the RFFO procedure while improving the gaps of the intermediate solutions. A natural research direction is therefore to explore the impact of adding valid inequalities on both the computational time and the final gap obtained. This approach may indirectly benefit from the clustering approximation developed in this paper since the structure of the modified problem may be compatible with a wider range of possible cuts compared to its sequence-dependent counterpart.

Acknowledgement

The authors thank the two anonymous reviewers for their helpful comments. This work has been partially financed by VIF Software.

Data availability statement

The data that support the findings of this study are available from the corresponding author, Larroche, F, upon reasonable request.

References

Absi, Nabil, Boris Detienne, and Stéphane Dauzère-Pérès. 2013. "Heuristics for the multi-item capacitated lot-sizing problem with lost sales." *Computers & Operations Research* 40 (1): 264–272.

Absi, Nabil, and Safia Kedad-Sidhoum. 2007. "MIP-based heuristics for multi-item capacitated lot-sizing problem with setup times and shortage costs." *RAIRO - Operations Research* 41 (2): 171–192.

Absi, Nabil, and Safia Kedad-Sidhoum. 2008. "The multi-item capacitated lot-sizing problem with setup times and shortage costs." *European Journal of Operational Research* 185 (3): 1351–1374.

Absi, Nabil, and Safia Kedad-Sidhoum. 2009. "The multi-item capacitated lot-sizing problem with safety stocks and demand shortage costs." *Computers & Operations Research* 36 (11): 2926–2936.

Almada-Lobo, Bernardo, Diego Klabjan, Maria Antónia carravilla, and José F. Oliveira. 2007. "Single machine multi-product capacitated lot sizing with sequence-dependent setups." *International Journal of Production Research* 45 (20): 4873–4894.

Almada-Lobo, Bernardo, Diego Klabjan, Maria Antónia Carravilla, and José F. Oliveira. 2010. "Multiple machine continuous setup lotsizing with sequence-dependent setups." *Comput Optim Appl* 47 (3): 529–552.

Armas, Jesica de, and Manuel Laguna. 2020. "Parallel machine, capacitated lot-sizing and scheduling for the pipe-insulation industry." *International Journal of Production Research* 58 (3): 800–817.

Beraldi, Patrizia, Gianpaolo Ghiani, Antonio Grieco, and Emanuela Guerriero. 2008. "Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling prob-

lem with sequence-dependent set-up costs." *Computers & Operations Research* 35: 3644–3656.

Campello, R. J. G. B., and E. R. Hruschka. 2006. "A fuzzy extension of the silhouette width criterion for cluster analysis." *Fuzzy Sets and Systems* 157 (21): 2858–2875.

Clark, Alistair R., Reinaldo Morabito, and Eli A. V. Toso. 2010. "Production setup-sequencing and lot-sizing at an animal nutrition plant through atsp subtour elimination and patching." *Journal of Scheduling* 13 (2): 111–121.

Copil, Karina, Martin Wörbelauer, Herbert Meyr, and Horst Tempelmeier. 2017. "Simultaneous lotsizing and scheduling problems: a classification and review of models." *OR Spectrum* 39 (1): 1–64.

Fiorotto, Diego Jacinto, and Silvio Alexandre de Araujo. 2014. "Reformulation and a Lagrangian heuristic for lot sizing problem on parallel machines." *Annals of Operations Research* 217 (1): 213–231.

Fiorotto, Diego Jacinto, Jackeline del Carmen Huaccha Neyra, and Silvio Alexandre de Araujo. 2019. "Impact analysis of setup carryover and crossover on lot sizing problems." *International Journal of Production Research* 0 (0): 1–20.

Fleischmann, Bernhard, and Herbert Meyr. 1997. "The general lotsizing and scheduling problem." *Operations-Research-Spektrum* 19 (1): 11–21.

Florian, M., J. K. Lenstra, and A. H. G. Rinnooy Kan. 1980. "Deterministic Production Planning: Algorithms and Complexity." *Management Science* 26 (7): 669–679.

Gicquel, Celine, Michel Minoux, Yves Dallery, and Jean-Marie Blondeau. 2009. "A tight MIP formulation for the Discrete Lot-sizing and Scheduling problem with parallel resources." In *2009 International Conference on Computers & Industrial Engineering*, 1–6. IEEE.

Guimarães, Luis, Diego Klabjan, and Bernardo Almada-Lobo. 2014. "Modeling lotsizing and scheduling problems with sequence dependent setups." *European Journal of Operational Research* 239: 644–662.

Helber, Stefan, and Florian Sahling. 2010. "A fix-and-optimize approach for the multi-level capacitated lot sizing problem." *International Journal of Production Economics* 123 (2): 247–256.

James, Ross J. W., and Bernardo Almada-Lobo. 2011. "Single and parallel machine capacitated lotsizing and scheduling: New iterative MIP-based neighborhood search heuristics." *Computers & Operations Research* 38: 1816–1825.

Jans, Raf, and Zeger Degraeve. 2008. "Modeling industrial lot sizing problems: a review." *International Journal of Production Research* 46 (6): 1619–1643.

Kaczmarczyk, Waldemar. 2013. "Modelling Set-up Times Overlapping Two Periods in the Proportional Lot-Sizing Problem with Identical Parallel Machines." *DMMS* 7 (1-2): 43.

Karimi, B., S.M.T. Fatemi Ghomi, and J.M. Wilson. 2003. "The capacitated lot sizing problem: a review of models and algorithms." *Omega* 31 (5): 365–378.

Karp, Richard M. 1972. "Reducibility among Combinatorial Problems." In *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, edited by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger, The IBM Research Symposia Series, 85–103. Boston, MA: Springer US.

Krarup, Jakob, and Ole Bilde. 1977. "Plant location, Set Covering and Economic Lot Size: An 0 (mn)-Algorithm for Structured Problems." In *Numerische Methoden bei Optimierungsaufgaben Band 3: Optimierung bei graphentheoretischen und ganzzahligen Problemen*, edited by L. Collatz, G. Meinardus, and W. Wetterling, 155–180. Basel: Birkhäuser.

Laan, Mark Van der, Katherine Pollard, and Jennifer Bryan. 2003. "A new partitioning around medoids algorithm." *Journal of Statistical Computation and Simulation* 73 (8): 575–584.

Lang, Jan Christian, and Zuo-Jun Max Shen. 2011. "Fix-and-optimize heuristics for capacitated lot-sizing with sequence-dependent setups and substitutions." *European Journal of Operational Research* 214 (3): 595–605.

Loparic, Marko, Yves Pochet, and Laurence A. Wolsey. 2001. "The uncapacitated lot-sizing problem with sales and safety stocks." *Mathematical Programming* 89 (3): 487–504.

Mateus, Geraldo R., Martín G. Ravetti, Maurício C. de Souza, and Taís M. Valeriano. 2010. "Capacitated lot sizing and sequence dependent setup scheduling: an iterative approach for integration." *Journal of Scheduling* 13 (3): 245–259.

Quadt, Daniel, and Heinrich Kuhn. 2008. "Capacitated lot-sizing with extensions: a review." *4OR* 6 (1): 61–83.

Robinson, E. Powell, and F. Barry Lawrence. 2004. "Coordinated Capacitated Lot-Sizing Problem with Dynamic Demand: A Lagrangian Heuristic." *Decision Sciences* 35 (1): 25–53.

Toso, Eli A. V., Reinaldo Morabito, and Alistair R. Clark. 2009. "Lot sizing and sequencing optimisation at an animal-feed plant." *Computers & Industrial Engineering* 57 (3): 813–821.

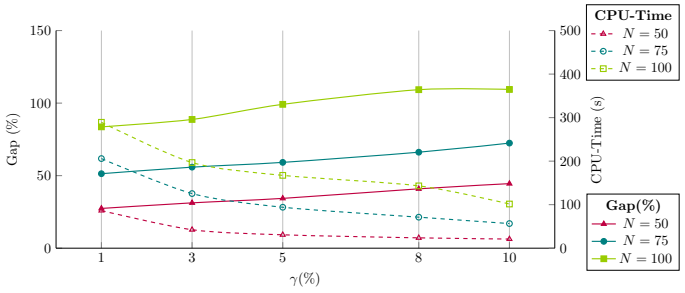
Wagner, Harvey M., and Thomson M. Whitin. 1958. "Dynamic Version of the Economic Lot Size Model." *Management Science* 5 (1): 89–96.

Xiao, Jing, Huasheng Yang, Canrong Zhang, Li Zheng, and Jatinder N.D. Gupta. 2015. "A hybrid Lagrangian-simulated annealing-based heuristic for the parallel-machine capacitated lot-sizing and scheduling problem with sequence-dependent setup times." *Computers & Operations Research* 63: 72–82.

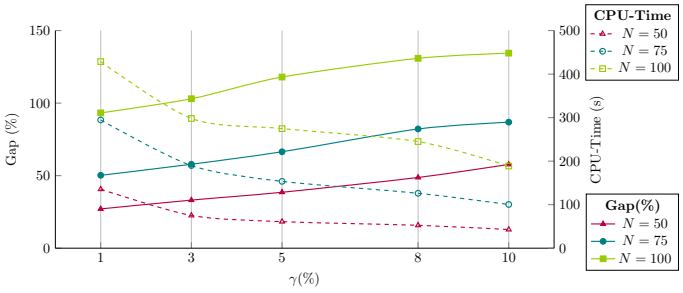
Xiao, Jing, Canrong Zhang, Li Zheng, and Jatinder N. D. Gupta. 2013. "MIP-based fix-and-optimise algorithms for the parallel machine capacitated lot-sizing and scheduling problem." *International Journal of Production Research* 51 (16): 5011–5028.

Appendix A. Analysis of the stop criteria in RFFO

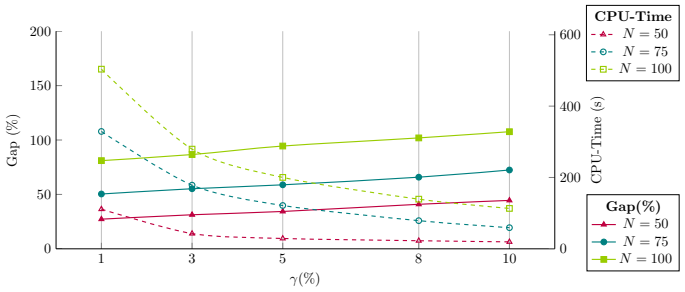
A.1. *Analysis of the gap limit γ*



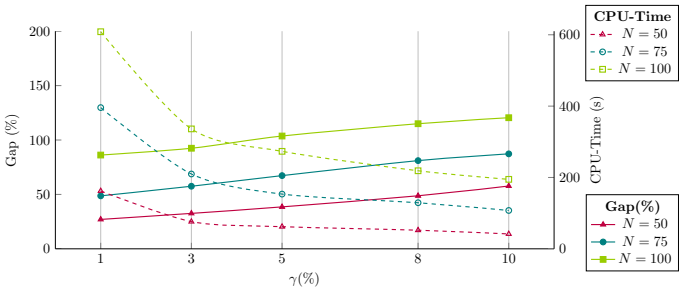
(a) $\rho = 30,\ T = 15$



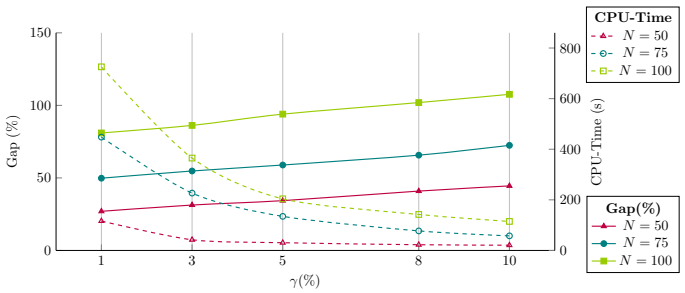
(b) $\rho = 30,\ T = 30$



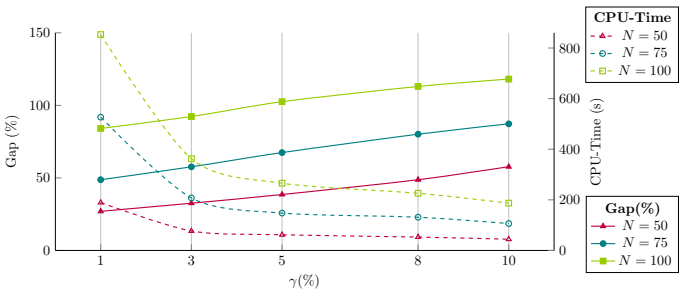
(c) $\rho = 60,\ T = 15$



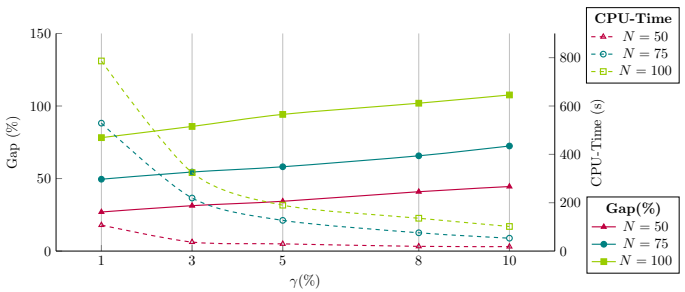
(d) $\rho = 60,\ T = 30$



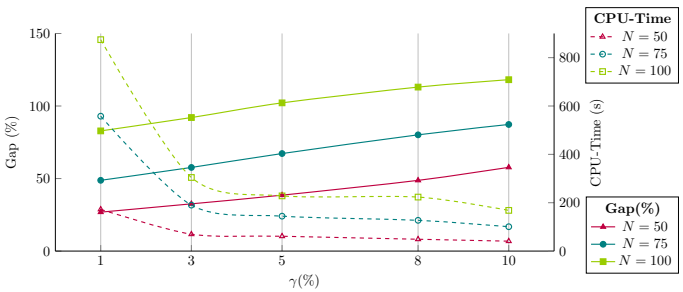
(e) $\rho = 90,\ T = 15$



(f) $\rho = 90,\ T = 30$

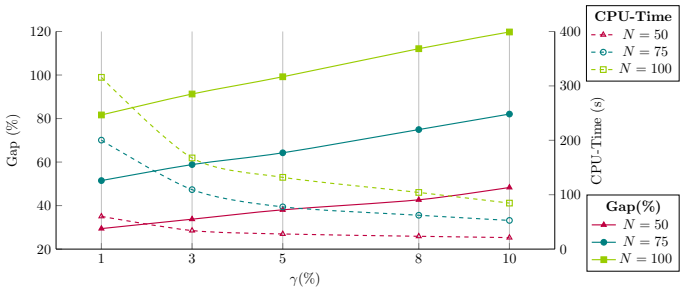


(g) $\rho = 120,\ T = 15$

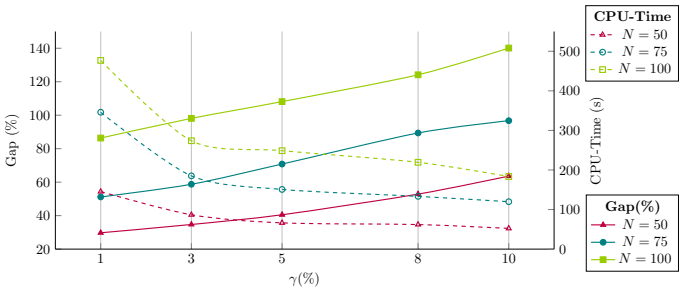


(h) $\rho = 120,\ T = 30$

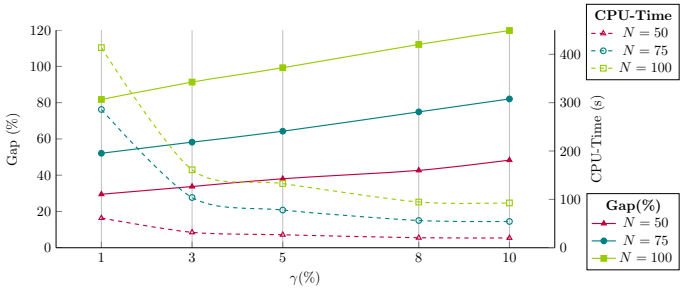
Figure A1. Results for RFFO with $\delta = 3$, $\sigma = 1$



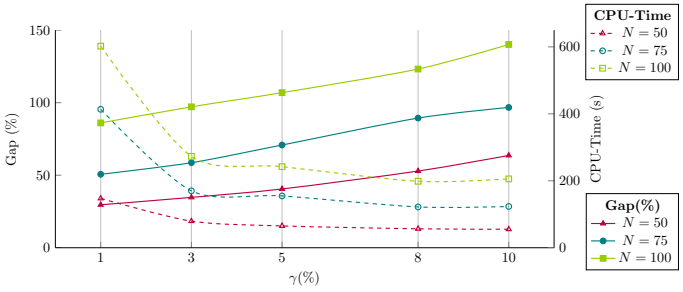
(a) $\rho = 30,\ T = 15$



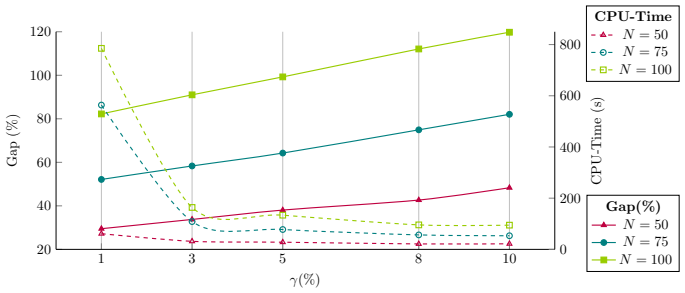
(b) $\rho = 30,\ T = 30$



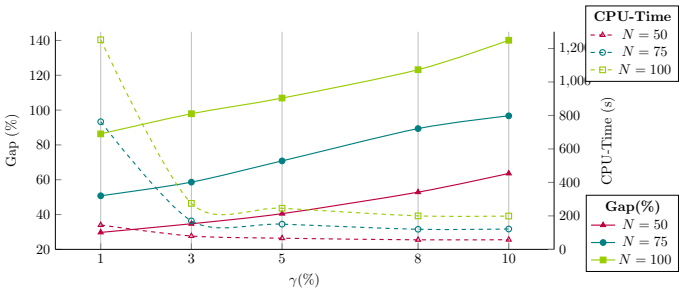
(c) $\rho = 60,\ T = 15$



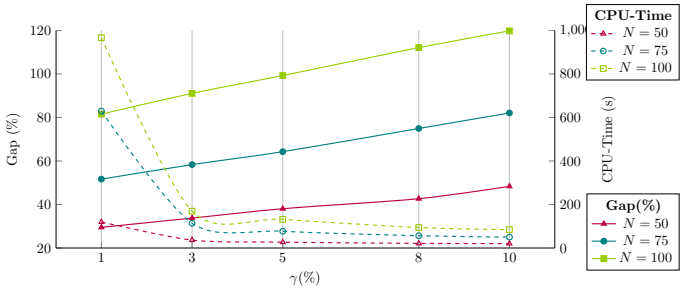
(d) $\rho = 60,\ T = 30$



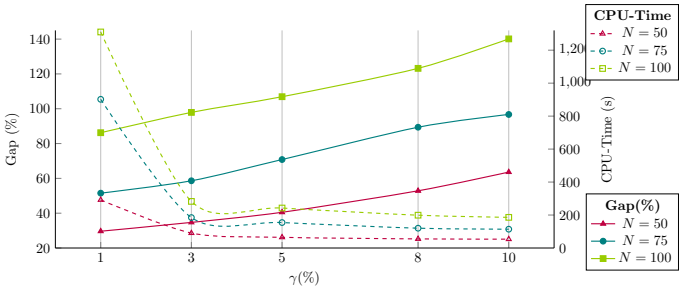
(e) $\rho = 90,\ T = 15$



(f) $\rho = 90,\ T = 30$



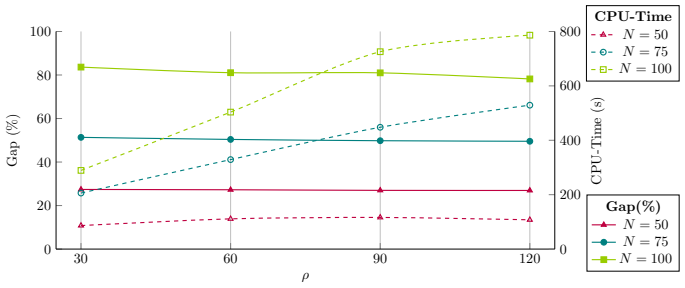
(g) $\rho = 120,\ T = 15$



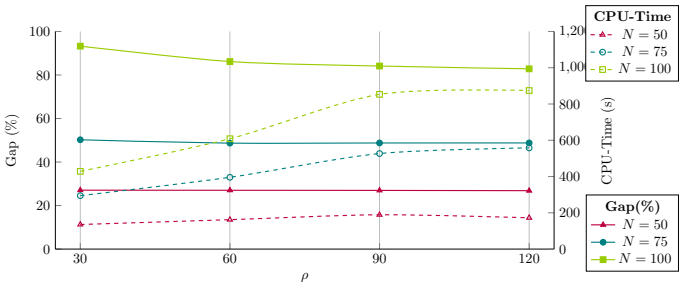
(h) $\rho = 120,\ T = 30$

Figure A2. Results for RFFO with $\delta = 2$, $\sigma = 1$

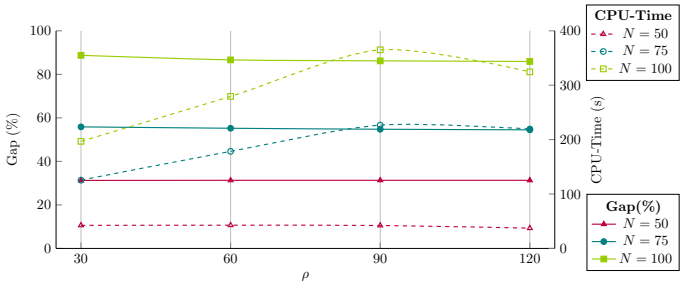
A.2. *Analysis of the time limit ρ*



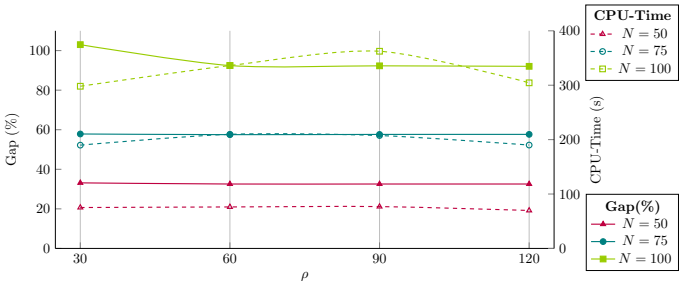
(a) $\gamma = 1,\ T = 15$



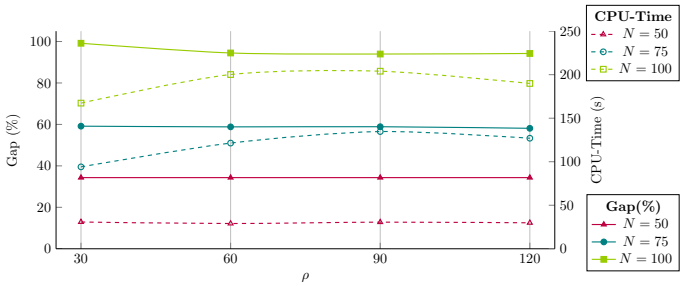
(b) $\gamma = 1,\ T = 30$



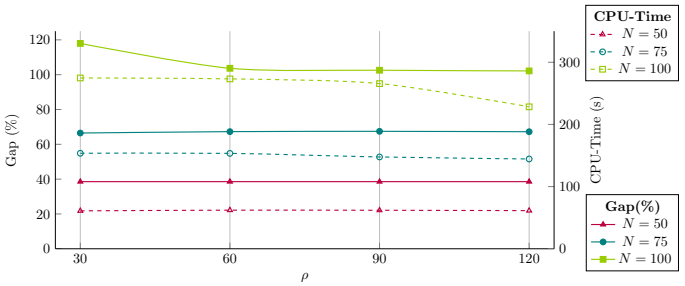
(c) $\gamma = 3,\ T = 15$



(d) $\gamma = 3,\ T = 30$

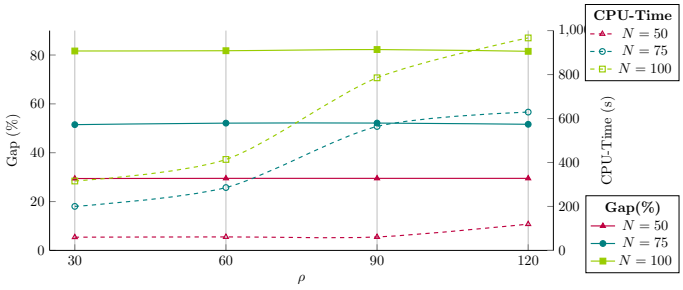


(e) $\gamma = 5,\ T = 15$

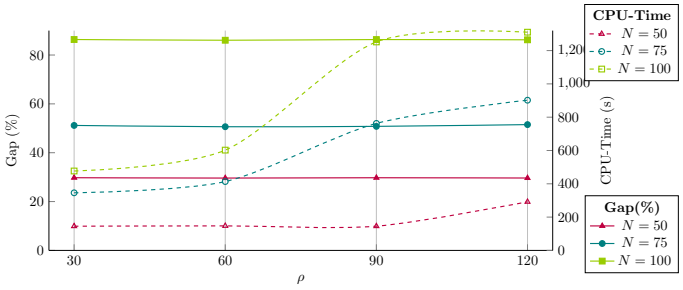


(f) $\gamma = 5,\ T = 30$

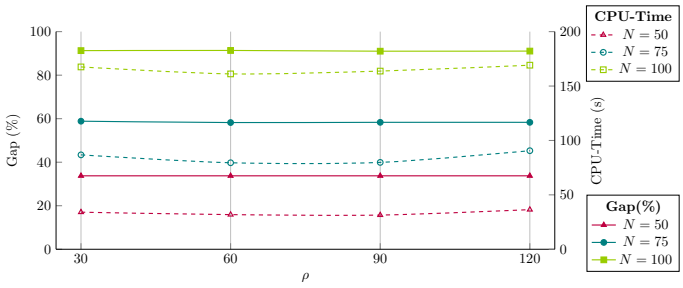
Figure A3. Results for RFFO with $\delta = 3$, $\sigma = 1$



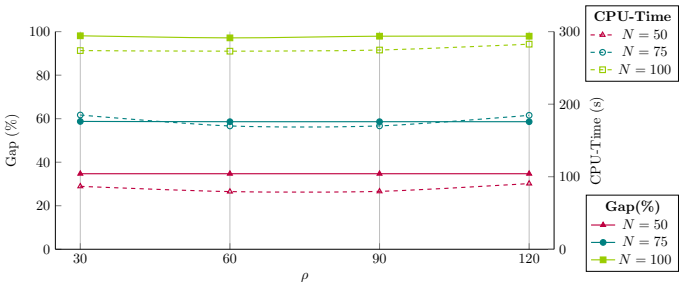
(a) $\gamma = 1,\ T = 15$



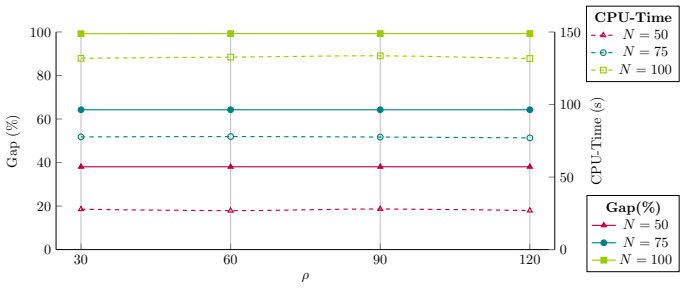
(b) $\gamma = 1,\ T = 30$



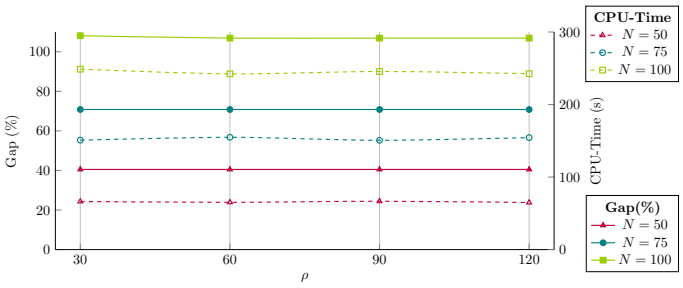
(c) $\gamma = 3,\ T = 15$



(d) $\gamma = 3,\ T = 30$



(e) $\gamma = 5,\ T = 15$



(f) $\gamma = 5,\ T = 30$

Figure A4. Results for RFFO with $\delta = 2$, $\sigma = 1$

Appendix B. Performances of RFFO for various combinations of parameters

Table B1. Computational results for RFFO ($\sigma = 1$, $\rho = 60$s)

Problem Size		20-1-15	20-1-30	20-2-15	20-2-30	30-1-15	30-1-30	30-2-15	30-2-30	40-1-15	40-1-30	40-2-15	40-2-30
RFFO(2, 1, 1, 60)	Avg. Gap (%)	12.69	7.68	6.08	12.0	7.78	16.69	8.48	8.13	17.39	25.13	10.07	11.6
	Sol. Time (s)	6.83	18.0	9.25	28.0	8.67	26.58	25.17	50.25	19.25	92.25	23.5	115.08
RFFO(2, 1, 3, 60)	Avg. Gap (%)	15.31	11.82	8.64	16.01	9.95	20.7	11.0	13.74	19.31	30.08	13.64	16.19
	Sol. Time (s)	4.83	13.67	6.58	20.0	6.42	19.17	13.83	32.83	13.67	46.92	14.33	57.17
RFFO(3, 1, 1, 60)	Avg. Gap (%)	12.26	7.46	5.68	11.17	6.57	16.26	7.89	7.36	14.85	23.27	9.35	10.82
	Sol. Time (s)	5.75	16.58	8.42	22.92	9.17	20.83	41.25	57.42	28.5	108.75	37.0	140.67
RFFO(3, 1, 3, 60)	Avg. Gap (%)	14.07	10.48	7.89	15.03	9.18	19.13	9.73	11.1	17.71	28.09	12.19	14.8
	Sol. Time (s)	4.17	9.67	4.92	13.58	4.75	13.67	15.33	26.75	11.58	44.83	15.75	49.42

Problem Size		50-1-15	50-1-30	50-2-15	50-2-30	75-2-15	75-2-30	75-4-15	75-4-30	100-2-15	100-2-30	100-4-15	100-4-30
RFFO(2, 1, 1, 60)	Avg. Gap (%)	26.08	25.06	10.86	13.38	38.66	43.47	25.15	22.02	66.59	59.26	33.74	41.5
	Sol. Time (s)	36.0	98.08	45.58	124.5	81.33	237.67	239.0	311.67	178.17	348.58	317.67	401.5
RFFO(2, 1, 3, 60)	Avg. Gap (%)	28.75	29.42	14.27	17.34	43.09	48.44	28.86	27.6	74.01	67.6	38.21	48.03
	Sol. Time (s)	18.83	50.08	20.75	63.92	35.25	105.83	93.17	144.33	78.75	180.0	112.92	194.92
RFFO(2, 1, 5, 60)	Avg. Gap (%)	32.25	32.78	16.97	24.05	47.14	56.46	33.29	37.91	78.97	73.44	43.3	55.9
	Sol. Time (s)	17.0	44.25	18.42	54.25	29.0	96.42	70.25	134.67	70.75	173.17	91.75	167.08
RFFO(3, 1, 1, 60)	Avg. Gap (%)	24.7	23.22	9.56	12.42	36.66	42.22	24.68	21.42	65.1	59.37	34.02	41.08
	Sol. Time (s)	48.5	87.0	85.08	122.0	105.17	254.42	285.08	274.08	289.83	386.25	278.92	381.83
RFFO(3, 1, 3, 60)	Avg. Gap (%)	27.29	26.61	12.46	16.93	40.03	48.29	27.69	26.26	69.92	63.1	36.37	46.03
	Sol. Time (s)	22.75	42.67	25.08	56.92	49.75	106.58	146.33	163.42	140.75	208.08	168.58	212.25
RFFO(3, 1, 5, 60)	Avg. Gap (%)	29.41	32.18	15.14	21.3	42.58	53.84	30.75	33.56	75.36	69.75	40.76	54.52
	Sol. Time (s)	16.5	35.33	18.5	48.08	29.75	86.83	107.67	120.92	95.83	175.5	127.42	166.33

Problem Size		125-4-15	125-4-30	125-6-15	125-6-30
RFFO(2, 1, 5, 60)	Avg. Gap (%)	70.73	66.47	57.09	71.14
	Sol. Time (s)	142.67	206.25	171.92	457.67
RFFO(2, 1, 8, 60)	Avg. Gap (%)	76.09	78.28	65.43	85.75
	Sol. Time (s)	110.5	178.5	133.17	380.25
RFFO(2, 1, 10, 60)	Avg. Gap (%)	87.49	92.15	73.03	101.18
	Sol. Time (s)	105.5	170.0	118.08	366.33
RFFO(3, 1, 5, 60)	Avg. Gap (%)	71.27	63.24	56.94	70.24
	Sol. Time (s)	155.08	199.75	200.83	448.75
RFFO(3, 1, 8, 60)	Avg. Gap (%)	76.02	72.59	63.29	78.78
	Sol. Time (s)	114.83	175.58	158.42	351.58
RFFO(3, 1, 10, 60)	Avg. Gap (%)	80.75	79.36	68.17	110.24
	Sol. Time (s)	109.75	167.92	139.25	344.08