Computers and Chemical Engineering 136 (2020) 106777

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Scheduling of flexible manufacturing plants with redesign options: A MILP-based decomposition algorithm and case studies

Natalia P. Basán[a], Mariana E. Cóccola[a,b], Alejandro García del Valle[c], Carlos A. Méndez[a,*]

[a] INTEC (UNL - CONICET), Güemes 3450, (3000) Santa Fe, Santa Fe, Argentina
[b] UTN-FRCU, Ing. Pereyra 676, (3260) Concepción del Uruguay, Entre Ríos, Argentina
[c] University of A Coruña, C/ Mendizábal s/n, Ferrol, 15403, Spain

ARTICLE INFO

Article history:
Received 23 October 2019
Revised 1 February 2020
Accepted 9 February 2020
Available online 13 February 2020

Keywords:
Scheduling problem
MILP model
Decomposition procedure
Redesign problem
Multipurpose units

ABSTRACT

In the last years, the operational research on scheduling problems has been moving away from rigorous optimization approaches into solution strategies being capable of returning practical and fast solutions for large-scale industrial problems. Following this line, this paper proposes a novel MILP-based decomposition procedure for solving scheduling problems arising in flexible manufacturing environments, which generally involve multipurpose units and assembly operations. The solution strategy also considers redesign constraints with the goal of improving the efficiency of the production system, preventing bottlenecks and balancing the equipment utilization. The proposal is validated through the resolution of several instances derived from three real-world case-studies coming from different industrial sectors. The computational results show that the decomposition procedure is capable of generating high quality solutions, sometimes the optimal one, with minimum computational effort for all problem instances considered.

1. Introduction

In the context of complex decision-making processes in industry, scheduling is defined as the problem of planning the manufacturing processes by allocating a set of operations to the available resources over a given time horizon taking into account some production targets (Pinedo, 2016; Castro et al., 2018). An efficient production schedule directly impacts on the productivity of the whole manufacturing process, allowing achieving a given objective according to the current needs of each industrial facility, such as the profit maximization or the minimization of the total cost, earliness and/or tardiness, changeover times or production downtime, or the reduction of makespan. Hence, in order to improve their viability in the current competitive environment, companies must invest in research and development, improving their production programming tools (Li and Ierapetritou, 2008; Harjunkoski et al., 2014; Aguirre and Papageorgiou, 2018); .

In the last two decades, the use of optimization techniques based on exact mathematical programming methods (LP, MILP, and MINLP), using different modeling concepts (Kondili et al., 1993; Pinto and Grossmann, 1995; Cerdá et al., 1997; Méndez et al., 2000, 2001), has played an important role in the field of scheduling research (Liu et al., 2010; Kopanos et al., 2011; Maravelias, 2012). Two extended reviews about scheduling problems and their solution methods can be found in Méndez et al. (2006) and Harjunkoski et al. (2014). This last one remarks that most of the solution approaches developed for solving scheduling problems are often presented in the literature from a modeling point of view and they are mainly tested only on small or medium sized problem instances. Due to the fact that all scheduling problems turn out to be NP-hard (Garey et al., 1976), it is unlikely that industrial instances can be solved efficiently through rigorous optimization approaches. Similarly, Georgiadis et al. (2019) underlines that a high interest from the Process System Engineering (PSE) community has been expressed for real-life industrial cases and consequently, several problem specific solutions have been generated for real industrial facilities. Nevertheless, these authors remark that there is still a significant gap between the academic research and the industrial practice, as only a few contributions have been successfully applied in real-life scheduling problems. Although the exact mathematical methods can model the operational constraints of complex production processes and efficiently solve small and medium-sized instances of them, such a solution technique is not able to efficiently represent complete industrial environments because of their high combinatorial complexity resulting in extremely large computational requirements.

* Corresponding author.
E-mail address: cmendez@intec.unl.edu.ar (C.A. Méndez).

https://doi.org/10.1016/j.compchemeng.2020.106777
0098-1354/

Nomenclature

Indices

i, i'	product
k	processing unit
s, s'	processing stage
t, t'	task
u	workstation

Sets

I	products
I^a	subset of products i that are components of final products
I^f	subset of products i that are final products
JT_i	subset of tasks t corresponding to product i
K	processing units
K_s	subset of processing units k that are suitable to perform operations of stage s
K_u	subset of processing units k integrating the workstation u
L	subset of units k that are released
S	processing stages
SA_i	subset of products i' that are subassemblies of product i
S^a	subset of processing stages s performing assembly operations
ST_s	subsets of tasks t corresponding to stage s
T	tasks
U	workstations or work cells
U_k	subset of workstations u compatible with the processing unit k

Parameters

pt_{is}	processing time of product order i at stage s
pt_t	processing time of task t
M	big number for big-M constraints
Seq_i	processing sequence of product i
N	quantity of products to be rescheduled at each iteration
N^{max}	maximum number of final products to be rescheduled simultaneously in an iteration
$lTime$	CPU time limit
$active_t$	determining if task t is active in the current iteration
$iter$	number of iterations
$BestSol$	saving the makespan of the best solution found
sMK	saving value of variable MK
sY_{tk}	saving values of variable Y_{tk}
$sW_{tt'}$	saving values of variable $W_{tt'}$

Continuous variables

MK	makespan
Ts_t	start time of task t
Tf_t	completion time of task t
V_k	taking a value greater than zero when processing unit k is used
X_{ks}	taking a value greater than zero when unit k is selected to perform some operation of stage s
X_{ku}	taking a value greater than zero when unit k is relocated in workstation u

Binary variables

$W_{tt'}$	determining if task t is processed before task t'
Y_{tk}	defining if task t is processed in processing unit k

Inspired by the industrial challenges and the gap reduction between the theoretical mathematical models and their industrial applicability, the researchers from PSE community have focused on the development of a wide range of different solution strategies, such as heuristics, meta-heuristics, decomposition-based approaches, or hybrid methods. The aim of these approaches is to compute "good" solutions for large-size problems within acceptable computational times. Particularly, heuristic and meta-heuristic approaches have been widely used to solve a variety of scheduling problems (Dauzère-Pérès and Paulli, 1997; Pranzo et al., 2003; Rossi, 2014) and even, they are becoming increasingly used due to their sill for solving many optimization problems. However, these solution approaches are usually tailor-made and cannot systematically estimate the degree of quality of the solution generated (Kopanos et al., 2010; Harjunkoski and Bauer, 2017).

On the other hand, solution strategies with complementary strengths can be mixed resulting in hybrid methods (Harjunkoski et al., 2014). This solution technique arises as the combination of two or more exact methods, as well as combinations of exact and heuristic methods. For instance, Maravelias (2006) proposed a hybrid method combining a MIP model with a heuristic scheduling algorithm for solving a multistage scheduling problem. Another type of hybrid approach generally used involves mathematical programming, simulation models, and heuristic techniques. When the scheduling problems are solved through discrete-event simulation model is needed to use dispatching rules, which are constructive heuristics for ordering the queue of jobs that are waiting in front of a resource. Following this line, there are relevant articles, such as those ones proposed by Basán et al. (2015), Cebral-Fernandez et al. (2016), and Basan et al. (2018). These authors developed discrete event simulation models, both to maximize productivity and the use of a shipyard's resources and to minimize the total time of the shipbuilding process. Also, MILP-based hybrid methods have received special attention due to the poor solutions given by the pure MILP models when they are applied in a monolithic way. Some approaches combine MILP techniques and event discrete simulation models, such as that one proposed by Castro et al. (2011), which presented a hybrid optimization and simulation-based method to reduce the computational burden involved in large-scale scheduling problem. In the same research line, Basán and Méndez (2016) proposed a hybrid algorithm based on MILP formulation, heuristic strategies and an advanced simulation model for solving the complex hoist scheduling problem. More recently, the same authors (Basán et al., 2017) introduced a novel hybrid simulation-based optimization method in order to generate complete schedules with efficient computational times for large-sized shipbuilding scheduling problems. The researchers have also begun to use the MILP techniques within decomposition algorithms in order to take advantage of their robustness and facilitate the resolution of the problems (Georgiadis et al., 2019). Usually in decomposition-based approaches, the problem at hand is solved through first determining an initial feasible solution and then, improving it through several rescheduling iterations. Some important contributions in this direction can be found in Kopanos et al. (2010), Aguirre et al. (2012), and Basán et al. (2019a, b, c).

The interest from the PSE community in developing efficient techniques for solving real-world industrial problems has opened a vast field of research. Chemical manufacturing facilities and other production environments can be classified as either sequential or network according to the material handling restrictions. In sequential processing, each batch/lot is characterized by a specific recipe, which determines the sequence of stages to follow while network facilities are characterized as more general and complex and have usually an arbitrary topology (Georgiadis et al., 2019). Sequential environments, in turn, can be categorized into three

subtypes according to the number of stages and the complexity of the production route (Maravelias, 2012). First, in a single-stage batch plant, each batch is processed in a single stage composing of one or more parallel units. Second, in a multistage batch plant, each lot is processed through a set of processing stages following the same recipe, similar to flexible flow-shop environments in discrete manufacturing. The third sequential environment is the multipurpose batch plant, where each batch has to be processed through a series of product-specific stages. According to Georgiadis et al. (2019), a facility is characterized as multipurpose when the routings are product-specific, or when a processing unit belongs to different processing stages depending on the product. The multipurpose facilities are equivalent to flexible job-shop problems in discrete manufacturing environments. Specifically, this paper relies on the development of an iterative algorithm for solving the integrated scheduling and redesign problem of multi-stage plants with assembly operations in flexible job shop environments, problem known in the literature as Flexible Job Shop Scheduling Problem with Assembly operation (FJSP-A). The definition of the Flexible Job Shop Scheduling Problem (FJSP problem) considers that an operation can be carried out on a set of compatible production units, thus a same unit can be suitable for operations in multiple processing stages. The units performing similar functions are grouped together in a work cell or workstation. In concordance with Shen et al. (2018), the availability of alternative units of performing similar operations can increase the performance and help to manage preventive maintenance or tackle breakdown and other unforeseen events.

From a scheduling point-of-view, chemical processes with a multipurpose environment are similar to flexible job-shop environments in discrete manufacturing systems. The scheduling problem for sequential chemical facilities is generally modelled in terms of batches and production stages, like jobs and operations in discrete manufacturing processes (Georgiadis et al., 2019). The FJSP problem is characterized to be NP-hard, i.e. its combinatorial complexity is exponentially increased with the number of jobs to be processed. Hence, no exact optimization approaches have been discovered so far to efficiently solve realistic instances of such a scheduling problem. In the last years, the FJSP problem has been subject of an exhaustive study and numerous techniques of different nature have been proposed for its resolution. For instance, Blazewicz et al. (1991) were the first to focus on developing mathematical formulations for FJSP problems, considering simple machines, parallel machines and job shop environments. Then, Pan (1997) compared different mathematical models for scheduling problems such as job shop and flow shop. Lately, Xia and Wu (2005) proposed a hierarchical solution approach for solving the multi-objective FJSP, while Fattahi et al. (2007, 2009) showed that hierarchical algorithms have better performance than integrated approaches. More recently, Lee and Maravelias (2017a) developed two mathematical models for simultaneous batching and scheduling in multipurpose batch plants with storage constraints. In addition, the same authors (Lee and Maravelias, 2017b) presented a new discrete-time mixed-integer linear programming formulations for short-term scheduling in multipurpose batch plants, using State-Task Network and Resource-Task Network representations. Other authors, such as Demir and Kürşat Işleyen (2013) and Roshanaei et al. (2013), also proposed mixed-integer linear mathematical programming models for solving FJSP problems. However, due to the complexity of the problem, the efforts are usually focused on heuristic methods and metaheuristics that, despite not guaranteeing the optimality of the solution, usually provide efficient solutions in acceptable computation time.

In this paper, we face with the integrated scheduling and redesign problem for multi-stage multipurpose plants. Particularly, once the best schedule that minimizes the makespan is computed in a first stage, the solution technique fixes the value of makespan and determines if the manufacturing plant is oversized through the minimization of the number of units utilized at each processing stage. In addition, the algorithm is able to relocate the units released to other compatible work cells in order to improve even more the efficiency of the production system through preventing bottlenecks and balancing the equipment utilization. Although the problem of redesigning multi-stage plants can be found in several industrial environments, to the best of our knowledge, this subject has received little or no attention at all in the literature. Most of the contributions have been focused on the design problem that conceptually involves the determination of both the size of storage tanks and processing units and the network structure of a system that is not yet operative. The integrated problem of scheduling and design for multi-stage multiproduct plants has been treated by Fumero et al. (2012). More recently, a hybrid metaheuristic to address the problem of multiproduct chemical batch plant design with parallel production lines has been presented by Verbiest et al. (2019a), where they use a MILP model and an iterated local search metaheuristic. Then, the same authors presented a RTN formulation and three decomposition approaches for solving the design and scheduling problem in the same type of production line (Verbiest et al., 2019b). Furthermore, other contributions on multipurpose plants were developed for solving realistic scheduling problems in an specific way because of the high complexity involved by each industrial case considered by the researchers Lin and Floudas (2001), Castro et al. (2005), Corsano et al. (2007), and Chibeles-Martins et al. (2010). All these proposals do not allow evaluating possible reconfigurations of the equipment or processing units in plants that are already operating normally and with a predefined configuration.

To face with the integration of both problems, scheduling and redesign, this article presents a novel hybrid solution strategy that efficiently solves real-life industrial instances arisen in FJSP-A environments (see Fig. 1). The paper starts with the development of a MILP model based on the general precedence concept, which is then embedded within an iterative algorithm in order to address large-scale scheduling problems arising in flexible manufacturing plants. The main goal is to minimize the makespan while satisfying all process constraints. The procedure first solves the scheduling problem considering the current plant configuration. Then, the algorithm runs again to determine if the plant is oversized by minimizing the number of units to be utilized, without worsen the makespan found in the first step. The units released can be reassignment to other compatible workstations. As a result, the algorithm determines the best plant configuration, detailing: (i) which units are used at each processing stage and (ii) which operations are performed by each equipment, including the starting and finishing times of each operation on each unit. The robustness and applicability of the proposed MILP-decomposition algorithm are demonstrated by developing several computational experiments on three real-world case-studies coming from different industrial sectors and featuring different sizes and characteristics. Also, the paper presents a comparative analysis of the computational efficiency exhibited by both the exact method via MILP formulation and the iterative method based on the decomposition algorithm.

The remaining of this manuscript is organized as follows. The problem definition is described in Section 2. The precedence-based MILP mathematical formulation with redesign constraints is proposed in Section 3 to model explicitly the FJSP problem. In Section 4, the iterative MILP-based algorithm is explained in detail. Numerical tests over one illustrative example and three case studies are then presented in Section 5. Finally, the concluding remarks are given in Section 6.

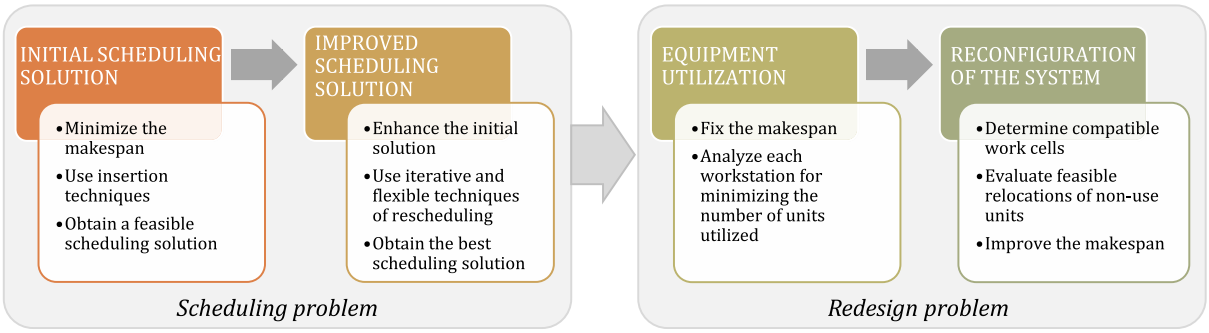


Fig. 1. Hybrid solution strategy for real-life industrial instances of FJSP-A environments.

2. Problem statement

The integrated problem of scheduling and redesign for flexible manufacturing plants involving multipurpose units and assembly operations can be mathematical modeled considering the following main assumptions:

- There is a set of products $I = \{i_1, i_2, .., |I|\}$ that are processed by following a predefined sequence of processing stages, not necessarily using all stages $s \in S$.
- There is a set of units $K = \{k_1, k_2, .., |K|\}$ allocated in groups or workstations u according to their suitability for performing operations of the same subset of processing stages.
- Each workstation $u \in U$ is composed of a subset of specific processing units $k \in K_u$, which can perform operations of one or more processing stages.
- The set $K_s \subset K$ determines the subset of units that are suitable to handle operations of stage s.
- No resource constraints except for equipment availability are considered. An equipment unit $k \in K$ cannot process more than one product at a time.
- When there are assembly operations, each final product $i \in I^f$ is created by assembling other intermediate products $i \in I^a$. I^f and I^a are subsets of set I, i.e. $I^f \subset I$ and $I^a \subset I$. The subset SA_i contains all subassemblies of product i.
- Assembly operations are performed by processing stages $s \in S^a$ ($S^a \subset S$).
- The assembly of a product $i \in I^f$ cannot start until all its corresponding parts (subassemblies) $i \in SA_i$ are finished and available.
- The processing time of product i at stage s, pt_{is}, is known in all cases.
- Transportation times between stages are considered negligible or included in the processing times, hence they are ignored.
- All model parameters, such as processing times, are deterministic.
- There are no setup or changeover times.
- Machine breakdowns and other unforeseen events that can interrupt the normal operation of the plant are not considered. Besides, preventive operations are not allowed during the programming horizon.
- Either UIS (unlimited intermediate storage) or NIS (non-intermediate storage) transfer policy between stages can be adopted.

Fig. 2 illustrates an example of a sequential multipurpose plant with assembly operations, where each product can follow different processing sequences. The products $i_1 - i_7$ can be processed in five work cells $u_1 - u_5$ to obtain four final products: i_6, i_8, i_9, and i_{10}. The workstation u_2 contains the units performing the assembly stage. For instance, the final product i_8 is obtained from joining products i_1 and i_7, while product i_6 does not pass by work cell u_2 because it does not require of assembling. Note that a specific path must be fulfilled to produce each final product. For instance, products i_2 and i_3 follow the production path $u_1 \rightarrow u_2 \rightarrow u_3 \rightarrow u_4 \rightarrow u_5$ to obtain final product i_9, while i_4 and i_5 follow the path $u_1 \rightarrow u_2 \rightarrow u_3 \rightarrow u_4$ to finally obtain i_{10}.

As mentioned above, a specific unit $k \in K$ is able to process only one product at a time and can be categorized as either single purpose or multipurpose. A single-purpose unit should be assigned to a single processing stage while multipurpose units can handle operations belonging to several stages. For treating with multipurpose units, the problem definition incorporates the following features:

- The set of tasks $T = \{t_1, t_2, \ldots, |T|\}$ defining all operations that must be scheduled.
- Each product $i \in I$ is decomposed in a subset of tasks $t \in JT_i$, where $JT_i \subset T$.
- In addition, the subset $ST_s \subset T$ includes all operations corresponding to stage s.
- For each product i that should be processed in a stage s ($s \in S_i$), there is a task $t \in (JT_i \cap ST_s)$.

A small example with 4 products i_1, i_2, i_3, i_4 and 2 processing stages s_1, s_2 is depicted in Fig. 3. In this case $|T| = 8$. Note that for the processing of product i_1, two tasks are defined: t_1 in stage s_1 ($t_1 \in ST_{s_1}$) and t_5 in stage s_2 ($t_5 \in ST_{s_2}$). At the same time, both tasks are part of the set JT_{i_1}. Therefore, each task is defined by a product and a stage.

On one hand, the scheduling problem has a goal the minimization of makespan (MK), which is a criterion based on the total time required to complete the processing of all products while operational constraints are satisfied. According to the production targets of the problem under study, other objectives, such as the maximization of profit, the minimization of the total cost, earliness and/or tardiness, can be also chosen for optimization. On the other hand, the redesign problem aims at determining the minimum number of units that can be utilized to accomplish with the objective function computed initially, and then evaluating feasible reassignments to further improve the original production target. Hence, the decision variables of the mathematical model are: *(i)* the assignment of task t to processing unit k through binary variable Y_{tk}, *(ii)* the sequencing of any pair of tasks (t, t') assigned to the same processing unit, determined by binary variable $W_{tt'}$, *(iii)* the assignment of some processing task to unit k or neither, defined by continuous variable V_k, and *(iv)* the relocation of non-use units k to compatible workstations u, X_{ku}.

3. Mathematical formulation

In this section, the problem addressed is mathematically represented as a Mixed Integer-Linear Programming (MILP) formu-

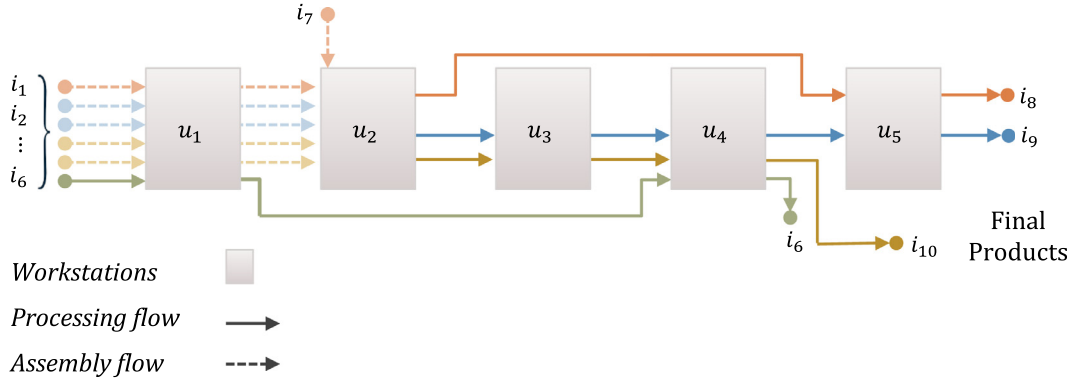


Fig. 2. Manufacturing process with multipurpose units and assembly operation (FJSP-A).

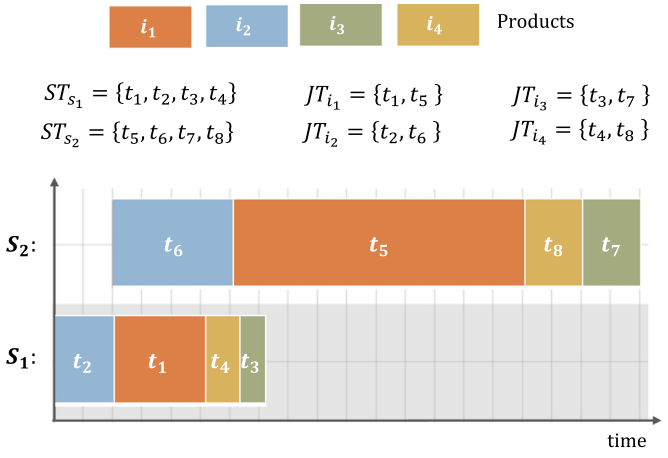


Fig. 3. Illustration of subsets JT_i and ST_s.

lation based on the general precedence concept presented by Méndez et al. (2000). The original proposal of these authors was developed for solving batch scheduling problems with dedicated units, without considering redesign constraints. The assignment and sequencing constraints presented initially in the cited paper were expressed in terms of batches and stages but here these constraints must be defined over the set of tasks T for effectively dealing with multipurpose units. Also, a new constraint is incorporated to the model in order to represent the assembly operations than can be carried out in discrete manufacturing systems. It is worth to remark that any other alternative formulation can be used to represent the scheduling problem addressed. The aim is to propose a robust formulation than can be taken as basis for the further development of the decomposition procedure. This section also presents a new set of constraints for mathematically representing the redesign problem. In the proposed mathematical formulation explained below, the constraints have been grouped according to their type: (i) scheduling decisions (assignment, timing, sequencing, and assembly) and (ii) redesign decisions (reallocation of units, rescheduling, and time adjustment).

3.1. Scheduling problem

- Allocation constraint

Eq. (1) represents the assignment constraint that is defined for every processing stage s and task $t \in ST_s$. This restriction determines that each task $t \in ST_s$ must be accomplished in just one processing unit $k \in K_s$, being K_s the subset of units k that can perform operations of stage s. Variable Y_{tk} represents the binary decision of whether to assign a task t to a unit k or not. Hence, Y_{tk} is active ($Y_{tk} = 1$) if task t is allocated to processing unit k; otherwise, the variable is set to zero.

$$\sum_{k \in K_s} Y_{tk} = 1 \qquad \forall s \in S, t \in ST_s \tag{1}$$

- Timing constraints

Constraint (2) computes the ending time of a task t, Tf_t, as its begin time Ts_t plus the processing time of t, which is known a priori through parameter pt_t. Note that $pt_t = pt_{is}$ when $t \in (JT_i \cap ST_s)$. Variables Tf_t and Ts_t are defined as continuous positive in the mathematical model.

$$Tf_t \geq Ts_t + pt_t \qquad \forall t \in T \tag{2}$$

In case of scheduling problems with due-date constraints, the variable Tf_t can be used for determining the earliness or tardiness for product i, being $t \in JT_i$.

Constraint (3) ensures that the processing of product i on any stage s cannot start until the processing of i in its previous stage has already been completed. The elements of subset of tasks JT_i are ordered according to the production recipe of product i.

$$Ts_t \geq Tf_{t'} \qquad \forall i \in I, (t, t') \in JT_i : t < t' \tag{3}$$

As a key feature in the most process industries is the handling of storage, Constraint (3) can be modified according to the storage policy adopted by the scheduler; it must be expressed as inequality for unlimited intermediate storage (UIS), while it becomes in equality when the non-intermediate storage (NIS) policy is adopted. In case of Zero-Wait scheduling policy, Constraints (2) and (3) should be expressed as equalities.

- Sequencing constraints

Constraints (4) and (5) are used to sequence any pair of tasks (t, t') assigned to a same processing unit k. These constraints are formulated as big-M constraints and follow the generalized precedence concept. The parameter M is an upper bound for the corresponding timing variables. According to the general precedence concept, the sequencing constraints on a unit k are defined for any pair of tasks (t, t') where $t < t', t \in ST_s, t' \in ST_{s'}$ and $k \in (K_s \cap K_{s'})$. The binary sequencing variable $W_{tt'}$. takes 1 as value when t is processed before than t'; otherwise, it is set to zero. Note that, if two tasks (t, t') are assigned to a unit k ($Y_{tk} + Y_{t'k} = 2$) and task t is chosen to be processed before task t', i.e. $W_{tt'} = 1$, then Eq. (4) forces that the begin time of t', $Ts_{t'}$, will be greater than the ending time of task t, Tf_t. However, if task t' is processed earlier than t, i.e. $W_{tt'} = 0$, the reverse statement is fulfilled and Eq. (5) becomes active.

$$Ts_{t'} \geq Tf_t - M(1 - W_{tt'}) - M(2 - Y_{tk} - Y_{t'k})$$

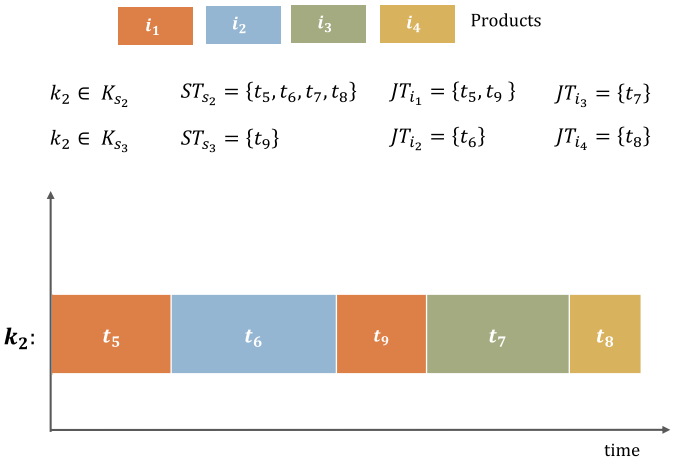


Fig. 4. A multipurpose unit performing operations of two processing stages.

$$\forall \left(s, s'\right) \in S, t \in ST_s, t' \in ST_{s'}, k \in (K_s \cap K_{s'}) : t < t' \tag{4}$$

$$Ts_t \geq Tf_{t'} - MW_{tt'} - M(2 - Y_{tk} - Y_{t'k})$$
$$\forall \left(s, s'\right) \in S, t \in ST_s, t' \in ST_{s'}, k \in (K_s \cap K_{s'}) : \; t < t' \tag{5}$$

In order to illustrate the definition of constraints (4) and (5) according to the generalized precedence concept, Fig. 4 shows a multipurpose unit k_2, in where five tasks of two different stages (s_2 and s_3) are assigned and sequenced. Note that t_5 and t_9 represent the processing of product i_1 at stage s_2 and s_3, respectively. The values of sequencing variables are: $W_{t_5,t_6} = 1$; $W_{t_5,t_7} = 1$; $W_{t_5,t_8} = 1$; $W_{t_5,t_9} = 1$; $W_{t_6,t_7} = 1$; $W_{t_6,t_8} = 1$; $W_{t_6,t_9} = 1$; $W_{t_7,t_8} = 1$; $W_{t_7,t_9} = 0$; $W_{t_8,t_9} = 0$.

In addition, some scheduling problems may require that the products i will be processed at any stage s following a predefined sequence, which is known a priori through parameter Seq_i. In this case, constraints (4) and (5) must be replaced by (6).

$$Tf_t \geq Tf_{t'} - M(2 - Y_{tk} - Y_{t'k}) \qquad \forall \left(i, i'\right) \in I, \; \left(s, s'\right) \in S, t \in JT_i,$$
$$t' \in JT_{i'}, t \in ST_s, t' \in ST_{s'}, k \in (K_s \cap K_{s'}) : Seq_i < Seq_{i'} \tag{6}$$

- Assembly constraint

In order to represent the assembly operations that can be performed on the manufacturing system, constraint (7) forces that the processing of product i in assembly stage $s \in S^a$ cannot begin until its subassemblies $i' \in SA_i$ have completed their processing in the previous stage $(s-1)$. The begin time Ts_t of task $t \in (JT_i \cup ST_s)$ must be greater than the end times $Tf_{t'}$ of tasks $t' \in (JT_{i'} \cup ST_{(s-1)})$.

$$Ts_t \geq Tf_{t'} \quad \forall s \in S^a, i \in I^a, i' \in SA_i, t \in JT_i, t' \in JT_{i'}, t \in ST_s, t' \in ST_{s-1} : s > 1 \tag{7}$$

- Objective function

The flexible manufacturing plants dealing with multipurpose units usually seek the effective utilization of their manufacturing resources, taking the machine versatility into consideration. A minimum makespan generally implies a high utilization of the processing units. Hence, the mathematical model aims at minimizing the makespan, which is computed through constraints (8) and (9).

$$Minimize\, MK \tag{8}$$

$$MK \geq Tf_t \qquad \forall t \in T \tag{9}$$

Due to there are many meaningful goals for choosing, the objective function (8) should be defined considering the production targets of the problem that is being solved. In case of considering other objective, such as the maximization of profit, the minimization of the total cost, earliness and/or tardiness, it is needed the incorporation of additional constraints in the mathematical formulation.

3.2. Redesign problem

When solving the original scheduling problem taking as goal the minimization of the makespan, or any other objective, the optimal solution may show that some workstations $u \in U$ have a low utilization rate, while other ones are the bottleneck. So, considering the existence of multipurpose units, it is possible to evaluate the relocation of some processing units $k \in K_u$ to other compatible workstations $u' \in U_k$ with the aim of preventing bottlenecks, balancing the equipment utilization and consequently, reducing the objective value given by the initial scheduling solution.

In the redesign stage, the first objective is to minimize the number of units to be utilized per workstation. After solving the MILP model (1)–(9), the variable MK is bounded by its optimal value (fixing it as upper bound) and the resulting model is solved again but considering two additional constraints (10) and (11). Here, the objective function is represented by constraint (10).

$$minimize \sum_{k \in K} V_k \tag{10}$$

$$Y_{tk} \leq V_k \qquad \forall t \in T, k \in K \tag{11}$$

V_k is a continuous positive variable that, according to Eq. (11), can be set to zero only when unit k is not utilized to perform any operation. In this case, it can be said that unit k is released and becomes part of the set L. Otherwise, if unit k is used to perform at least one task t, the variable V_k will take a value equal to 1 because it is included in the objective function (of minimization) as a positive term. The MIP solver will always assign the minimum feasible value to variable V_k, i.e. 0 or 1, depending on Eq. (11).

- Relocation of released units

To improve the efficiency of the production system, feasible reconfigurations of the facility can be evaluated. This option is modeled through Constraint (12), which establishes that each released unit $k \in L$ can be relocated at most to a single compatible workstation $u \in U_k$.

$$\sum_{u \in U_k} X_{ku} \leq 1 \qquad \forall k \in L \tag{12}$$

The decision to relocate a unit $k \in L$ to workstation u, i.e. $X_{ku} = 1$, implies that k will be able to perform the operations belonging to all stages $s \in S_u$. This condition is modeled through constraint (13).

$$X_{ku} = X_{ks} \qquad \forall u \in U_k, k \in L, s \in S_u \tag{13}$$

- Rescheduling tasks

Constraints (14) and (15) are used to assign the tasks t to be processed by each unit $k \in L$ relocated in a compatible workstation $u \in U_k$. Although all tasks $t \in T$ have already assigned to some processing unit in the initial scheduling solution, some of them can be rescheduled to any unit $k \in L$. In this case, the binary variable Y_{tk} is used to compute the reassignment of task t to released unit $k \in L$. It is important to mention that the reassignment of tasks to non-released units ($k \notin L$) is not allowed. In other words, $Y_{tk} = 0 \; \forall k \notin L, t \in T$. In addition, Constraint (15) determines that task $t \in ST_s$ can be assigned to unit $k \in L$ only in the case that such a unit is able to perform operations of stage s, i.e. $X_{ks} = 1$.

$$\sum_{k \in L} Y_{tk} \leq 1 \qquad \forall t \in T \tag{14}$$

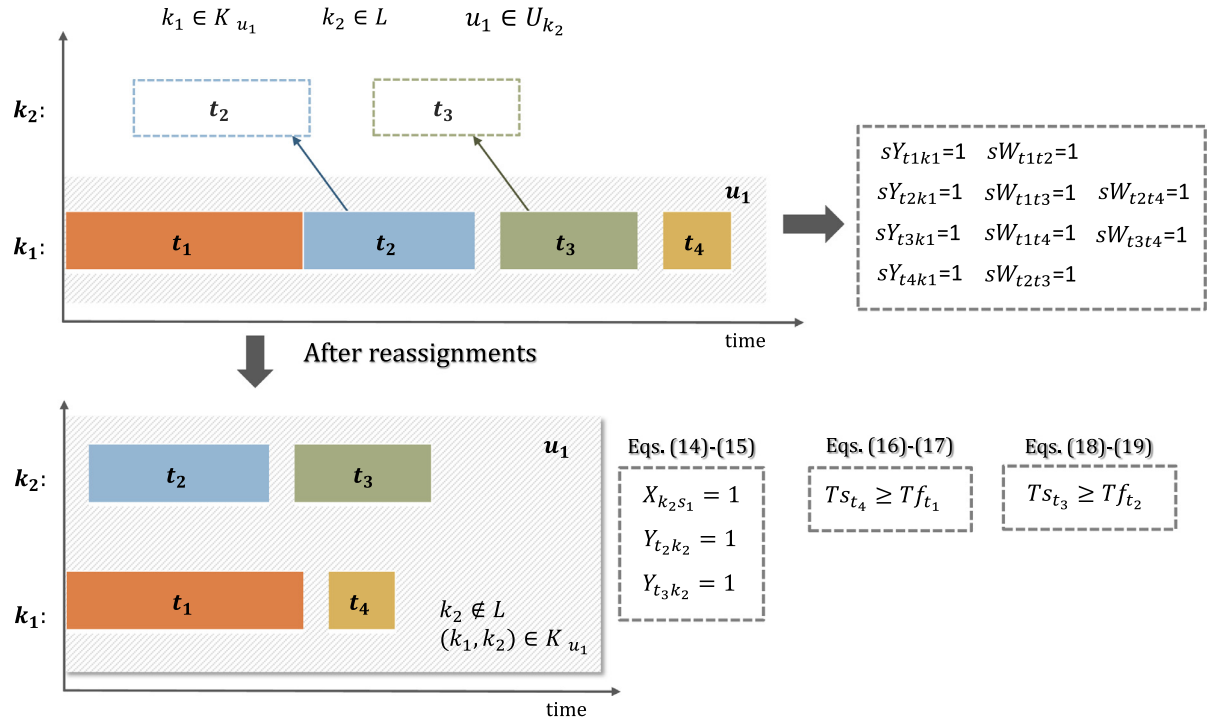


Fig. 5. Reassignment of tasks to released units and time adjustment.

$$Y_{tk} \leq X_{ks} \qquad \forall s \in S, t \in ST_s, k \in L \tag{15}$$

- Time settings

Once the reassignment actions have been performed on the initial schedule, it is needed to update the timing decisions for each task. On the one hand, Constraints (16) and (17) are used to adjust the begin time of tasks that are assignment to units $k \notin L$. Parameters $sW_{tt'}$ and $sY_{t'k}$ save the value of variables $W_{tt'}$ and $Y_{t'k}$ in the original scheduling solution, respectively.

$$Ts_{t'} \geq Tf_t - M\sum_{k' \in L} Y_{tk'}$$
$$\forall\ (t,t') \in T, k \notin L, sW_{tt'} = 1, sY_{tk} = 1, sY_{t'k} = 1 : t < t' \tag{16}$$

$$Ts_t \geq Tf_{t'} - M\sum_{k' \in L} Y_{tk'}$$
$$\forall (t,t') \in T, k \notin L, sW_{tt'} = 0, sY_{tk} = 1, sY_{t'k} = 1 : t < t' \tag{17}$$

On the other hand, constraints (18) and (19) are used to determine the sequencing decisions for each unit $k \in L$. For greater clarity to the reader, constraints (14)–(19) are illustrated in Fig. 5, wherein the unit $k_2 \in L$ is relocated to compatible workstation u_1 and thus, it can perform operations of this work cell, such as t_2 and t_3.

$$Ts_{t'} \geq Tf_t - M(1 - W_{tt'}) - M(2 - Y_{tk} - Y_{t'k}) \qquad \forall (t,t') \in T, k \in L : t < t' \tag{18}$$

$$Ts_t \geq Tf_{t'} - MW_{tt'} - M(2 - Y_{tk} - Y_{t'k}) \qquad \forall (t,t') \in T, k \in L : t < t' \tag{19}$$

4. The solution strategy

A full space approach, such as that one presented in the previous section, may not be suitable for solving realistic instances of industrial problems because their resolution through exact methods is either impossible or yield poor solutions with high integrality gaps. An efficient strategy for solving real-life industrial cases that takes advantage of the robustness exhibited by the exact optimization approaches without falling into the pure use of heuristics or meta-heuristics is to disaggregate the problem structure in a logical way and to resolve manageable model sizes at each iteration.

The general scheme of the decomposition algorithm proposed in this paper is depicted in Fig. 6, where two major procedures can be identified according to the problem to be solved: (i) the first stage deals with the scheduling problem, wherein an initial solution is computed and then it is gradually improved, considering the reallocation and reordering of tasks in order to obtain the best schedule; (ii) the other stage deals with the redesign problem, wherein the use and possible oversizing of multipurpose units in each workstation are evaluated for subsequent relocations, with the aim of improving the efficiency of the production system and schedule. Therefore, the solution of each stage is constructed iteratively through the resolution of the MILP model several times but considering a reduced search space in each solver execution. The procedure aims at converging to a high-quality solution with relatively low computational effort. The proposed algorithm is described in detail in the following subsections.

4.1. Scheduling stage

The first phase of the algorithm solves the scheduling problem by first constructing an initial feasible solution and then improving it through several rescheduling iterations. Thus, it sequentially performs two sub-stages.

4.1.1. Constructing an initial scheduling solution

The first step aims at finding an initial feasible schedule considering the current plant configuration. For this goal, any constructive heuristic can be used. The strategy used here is based on the sequential construction heuristic proposed by Kopanos et al. (2010), in which a subset of products are inserted

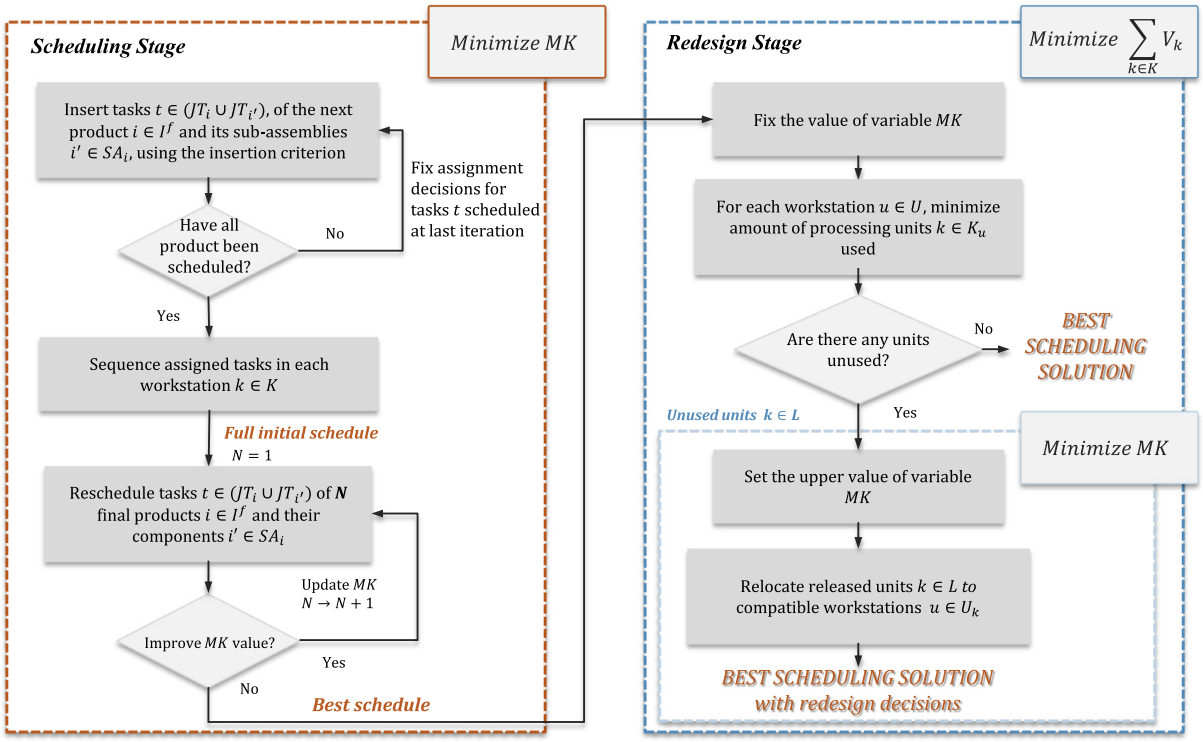


Fig. 6. General structure of the MILP-based algorithm.

```
SET   active_t = false;
FOR iter = 1 to |I^f|
    LOOP (i ∈ I^f and iter = Seq_i)
        LOOP (t ∈ JT_i)
            active_t = true;
        END LOOP;
        LOOP (i' ∈ SA_i)
            LOOP (t ∈ JT_i')
                active_t = true;
            END LOOP;
        END LOOP;
    END LOOP;
    Solve MILP model with constraints (1)-(3), (6)-(9)
    LOOP (t ∈ T and active_t = true)
        FIX binary variables Y_tk;
    END LOOP;
END FOR
Solve MILP model with constraints (1)-(9)
SAVE initial solution in parameters sMK, sY_tk and sW_tt'
```

Fig. 7. Pseudo-code of initial feasible solution phase.

at each iteration until all orders are finally inserted. The constructive heuristic used should be capable to give a feasible solution for the scheduling problem in few seconds of CPU time. The convergence to a feasible schedule should be reached with low computational effort, and in some cases, it can be viewed as a good approximation to the industrial reality. Therefore, it is needed to set a criterion for selecting the number of products to be inserted at each iteration and the order in which they will be inserted. In order to minimize the MIP solver search space and ensure the quick mathematical model resolution for every iteration, the easiest and most effective heuristic is to insert the products $i \in I^f$ one-by-one and select them following an insertion sequence, such as that one given by the parameter Seq_i. Note that when a final product $i \in I^f$ is selected to be inserted and scheduled in each iteration, its sub-assemblies $i' \in SA_i$ must also be considered for scheduling. With this insertion criterion, the number of iterations of the constructive stage is equal to $|I^f|$. The pseudo-code of the constructive heuristic is given in Fig. 7.

```
SET bestSol = sMK ; flag = true;
FOR N = 1 TO N^max
    WHILE flag
        flag = false;
        FOR iter = 1 TO (|I^f| − N + 1)
            active_t = false;
            LOOP (i ∈ I^f and i = iter)
                LOOP (t ∈ JT_i)
                    active_t = true;
                END LOOP;
                LOOP (i' ∈ SA_i)
                    LOOP (t ∈ JT_i')
                        active_t = true;
                    END LOOP;
                END LOOP;
            END LOOP;
            SOLVE MILP model (1)-(9)
            with a CPU time limit of lTime
            IF (MK < BestSol)
                bestSol = MK;
                flag = true;
                LOOP (t ∈ T and active_t = true)
                    sY_tk = Y_tk;
                END LOOP;
                LOOP ((t,t') ∈ T
                    and t < t' and (active_t = true or active_t' = true))
                    sW_tt' = W_tt';
                END LOOP;
            END IF;
        END FOR;
    END WHILE;
END FOR;
```

Fig. 8. Pseudo-code of iterative improvement phase.

A boolean parameter called $active_t$ is used by the algorithm to determine if a task t is scheduled at iteration *iter*. The model with constraints (1)–(3), (6)–(9) is solved at each iteration but considering only the tasks t with $active_t = true$. After each solver execution, the binary variables Y_{tk} are fixed and the procedure continues with the scheduling of the next task. Particularly, at each iteration, only assignment decisions are considered. As can be seen in the pseudo-code of Fig. 7, the sequencing decisions $W_{tt'}$ are determined by solving the MILP model with constraints (1)–(9) once all assignments decisions Y_{tk} have been fixed. Another feasible alternative is to determine both the assignment and sequencing decisions in a simultaneous way for all tasks selected for scheduling at each iteration. Once all tasks have been scheduled, the solution found (initial schedule) is saved in parameters *sMK*, sY_{tk}, and $sW_{tt'}$.

4.1.2. Improving the scheduling solution

In this phase, the initial scheduling solution found by the constructive heuristic is systematically improved by performing several rescheduling iterations. A rescheduling action consists of releasing a subset of products $i \in I^f$ from the current schedule in order to find for them better unit assignments or sequencing. As in the constructive stage, when a product i is selected for rescheduling it means that all tasks $t \in JT_i$ can be reassigned or resequenced. The same happen with tasks $t \in JT_{i'}$, where $i' \in SA_i$. The pseudo-code of the improvement stage is given in Fig. 8. At each iteration, the MILP model integrated by constraints (1)–(9) is solved but considering just the binary variables Y_{tk} and $W_{tt'}$ for the tasks t with the parameter $active_t = true$. For the remaining tasks t with $active_t = false$, only timing decisions can be taken and their assignment and sequencing decisions are fixed and known through parameters sY_{tk} and $sW_{tt'}$.

A key point to define in the improvement stage is the number of products $i \in I^f$ to be rescheduled at each model execution, given by parameter N. Let's consider a small problem with four products (i_1, i_2, i_3, and i_4). Fig. 9 shows the products rescheduled at each iteration for different values of parameter N: $N = 1$ to $N = |I|$. Note that the fewer the value of N, the higher the number of iterations and smaller the solver search space, resulting in manageable model sizes. However, as the value of N increases: (i) the number of iterations is reduced, (ii) the feasible region of the resulting mathematical model becomes larger, and hence, (iii) the resolution of the MILP model becomes more complex and intractable.

The algorithm developed in this paper is configured for executing the improvement stage several times starting with $N = 1$ and go on increasing this value by one until, either the maximum number of iterations N^{max} is achieved or the makespan is not improved. A CPU time limit (*lTime*) is imposed for each model execution. The values of parameters N^{max} and *lTime* should be defined from a previous analysis of the problem under resolution.

Every time that all products $i \in I^f$ and their subassemblies have been rescheduled, the procedure checks if the solution (makespan)

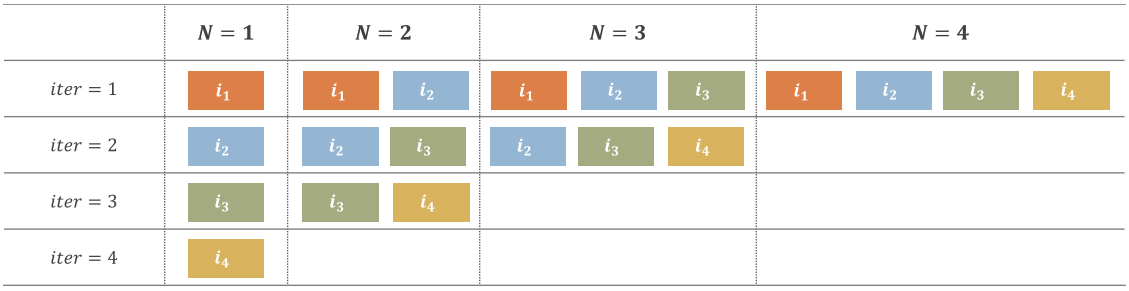


Fig. 9. Products rescheduled at each iteration for different values of *N*.

```
FIX variable MK = BestSol
FOR iter = 1 to |U|
    active_t = false ;
    LOOP (s ∈ S_u and u = iter)
        LOOP (t ∈ ST_s)
            active_t = true ;
        END LOOP;
    END LOOP;
    Solve MILP model (1)-(7), (9)-(11)
    LOOP (k ∈ K_u and u = iter)
        FIX positive variables V_k ;
        LOOP (t ∈ T and active_t = true)
            sY_tk = Y_tk ;
        END LOOP;
    END LOOP;
    LOOP ((t, t') ∈ T and t < t' and (active_t = true or active_t' = true))
        sW_tt' = W_tt' ;
    END LOOP;
END FOR;
```

Fig. 10. Pseudo-code of unit minimization phase.

is improved. If the solution found is better than the last one reported as the best (*BestSol*), the algorithm: (i) updates the best solution, (ii) saves the allocation and sequencing decisions in parameters sY_{tk} and sW_{tk}, and (iii) starts the rescheduled iterations again from the beginning. Otherwise, the procedure ends and reports the solution saved in parameters sY_{tk} and sW_{tk} as the best solution found.

4.2. Redesign stage

This algorithmic stage takes as input the best scheduling solution and evaluates the redesign of the plant by analyzing: (i) if some processing unit can be released through the reassignment of the tasks assigned to such a unit to other equipment, while maintaining the makespan value; and if so, (ii) the feasible relocations of such released units to other compatible workstations in order to improve the efficiency of the production system.

The redesign stage involves the resolution of mathematical model integrated by constraints (1)–(7) and (9)–(11) for minimizing the number of processing units to be utilized and then, the resolution of mathematical model (2), (3), (6)–(9), and (12)–(19) for possible reassignments of tasks to the released units with the goal of minimizing the original value of makespan. The resolutions of both models via a full space approach are computationally expensive when real-life industrial instances are solved. Therefore, the redesign stage is also decomposed to be iteratively solved as explained follows.

4.2.1. Analysis of the equipment utilization

This step aims at minimizing the number of processing units required to process all tasks $t \in T$ within the time given by the best solution found in the *Scheduling stage*. Once the best schedule that minimizes the makespan is reported by the previous step of the algorithm, the procedure fixes the value of variable *MK* and several rescheduling actions are iteratively applied on the current scheduling, by solving the MILP model with constraints (1)–(7) and (9)–(11) for each workstation $u \in U$. The pseudo-code for the first phase of the redesign stage is shown in Fig. 10.

The procedure iterates on each workstation $u \in U$ in order to minimize the number of units $k \in K_u$ utilized at each of them. Hence, at each iteration, the procedure activates all tasks $t \in ST_s$ that belong to stages $s \in S_u$. Only the tasks with $active_t = true$ can be reassignment. After each resolution of the MILP model, the algorithm fixes the value of variables V_k for all k involved in the current iteration. The variable V_k takes value 1 if k is used in the workstation u ($k \in K_u$), otherwise, V_k takes value 0 meaning that the unit k "is released" and becomes candidate for a future relocation. In this last case, the unit k becomes part of the set L ($k \in L$).

```
FIX variable MK = BestSol
FOR iter = 1 to |L|
      FREE variables Y_tk, X_ks, X_ku ;
      LOOP (k ∈ L and k = iter)
            LOOP (u ∉ U_k)
                  FIX variables X_ku = 0;
            END LOOP;
      END LOOP;
      LOOP (k ∈ L and k ≠ iter)
            FIX binary variables Y_tk = 0;
            FIX positive variables X_ks = 0;
      END LOOP;
      SOLVE MILP MODEL (2), (3), (6)-(9), (12)-(19)
      LOOP (k ∈ L and k = iter)
            LOOP (t ∈ T)
                  sY_tk = Y_tk;
            END LOOP;
            LOOP ((t,t') ∈ T and t < t' and (Y_tk = 1 or Y_t'k = 1)
                  sW_tt' = W_tt';
            END LOOP;
      END LOOP;
END FOR;
```

Fig. 11. Pseudo-code of reconfiguration phase.

The procedure ends when all workstation $u \in U$ have been evaluated. As in the previous algorithmic stages, allocation and sequencing decisions are saved in the parameters sY_{tk} and $sW_{tt'}$, respectively.

4.2.2. Reconfiguration of the production system

Once the set of unused processing units ($k \in L$) is determined, the relocation of these units to other possible workstations $u \in U_k$ is evaluated. The feasible reallocations should be defined for each specific problem, establishing the compatible workstations for each unit through set U_k. At first, it can be assumed that each unit $k \in L$ can be relocated to any workstation $u \in U$. It is important to highlight that the purpose of this last phase of the algorithm is to analyze the configuration of the production system and carry out an efficient relocation of all units previously released in order to minimize the makespan and improve utilization of workstations. Therefore, the best value of makespan reported by the *Scheduling stage* (*BestSol*) is set as the upper bound of the variable *MK*.

The relocation of released units can be taken simultaneously by solving the MILP model involving the constraints (2), (3), (6)–(9), and (12)–(19) with the goal of minimizing the makespan. However, for some complex instances, the resolution of this full mathematical model can be very expensive from the computational requirement point of view. In these cases, it is possible to use the relocating strategy of the units $k \in L$ one-by-one, in an iteratively way, as shown the pseudo-code given in Fig. 11. At each iteration, the assignment and sequencing decisions are activated only for the unit $k \in L$ under consideration. For the remaining units $k \in L$, the variables Y_{tk} and X_{ks} are set to zero. At end of each iteration, the current schedule is update with the reassignments of tasks to the unit $k \in L$ under iteration, saving the solution in parameters sY_{tk} and $sW_{tt'}$. The procedure ends when all units $k \in L$ have been iterated.

5. Case studies and computational results

When industrial problems are solved through an optimization approach, both the computational time and the quality of the solution found depend on both the quantity of products and the number of processing units involved in the problem configuration. In this way, the applicability and efficiency of the proposed solution strategy are tested by solving different problem sizes derived from three real-world case studies. The aim is to evaluate and illustrate the performance, flexibility and scalability of the algorithm through the resolution of different problem instances. Firstly, the algorithm is verified by solving a small illustrative example arising in a flexible manufacturing system that accounts for assembly operations and multipurpose units. Then, several instances derived from three real-world cases studies coming from different industrial sectors are solved by using both the exact optimization approach and the decomposition procedure. First, a real-world case study coming from the pharmaceutical industry is tackled. The second case study deals with the scheduling problem of an automotive company processing parts of automobiles. Finally, the third real-life example arises in a shipping company dedicated to the construction of large ships. The experimentation developed allows comparing the performance of both solution strategies in terms of solution quality and computational burden, estimating an approximated measure of the quality of the solutions provided by the decomposition algorithm.

The algorithm and underlying models were codified in GAMS 25.1.2 using CPLEX 12.6.3 as the MILP solver, which is set for running in a parallel deterministic mode using up to twelve threads. The hardware is a DELL PRECISION T5500 Workstation with six-core Intel Xeon processor (2.67 GHz) and 8 GB of RAM. Either a relative optimality of tolerance of zero or a time limit of 1 CPU hour was imposed as the termination criteria on the resolution

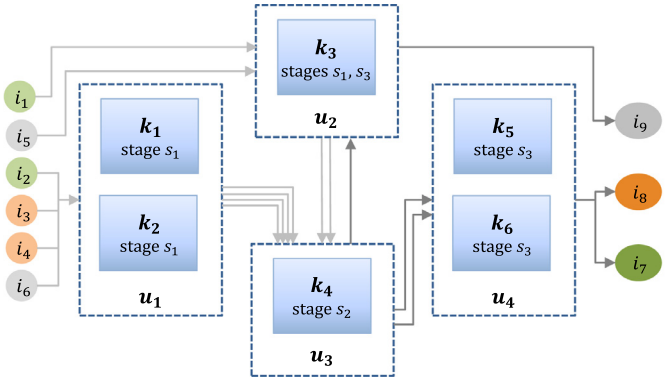


Fig. 12. Flowchart of the illustrative example.

Table 1
Processing times [hours] for the illustrative example.

Product	Stages		
	s_1	s_2	s_3
i_1	4	–	–
i_2	5	–	–
i_3	5	–	–
i_4	8	–	–
i_5	3	–	–
i_6	9	–	–
i_7	–	9	10
i_8	–	4	8
i_9	–	7	6
Units	k_1, k_2, k_3	k_4	k_3, k_5, k_6

of every example solved by the exact optimization approach. For those problem instances whose solutions featuring a high integrality GAP after 1 h of computational time, the MILP model is solved again but considering a longer CPU time limit in order to evaluate the reduction of integrality GAP to a lower value or eventually the convergence to the optimal solution.

5.1. A small illustrative example

In order to illustrate the characteristics of the scheduling problem in flexible manufacturing systems, a toy example is presented and solved in this section. Such an example involves the processing of $|I| = 9$ products. Three of them are final products ($|I^f| = 3$), each one composed by two subassemblies ($|I^a| = 6$). Specifically, $SA_{i_7} = \{i_1, i_2\}$, $SA_{i_8} = \{i_3, i_4\}$, and $SA_{i_9} = \{i_5, i_6\}$. The production facility consists of 6 equipment units ($|K| = 6$) for carrying out 3 processing stages ($|S| = 3$). As shown Fig. 12, there are 4 workstations ($|U| = 4$) and those units that can perform operations belonging to the same subset of processing stages are grouped together in the same workstation u. Note that there is an only multipurpose unit k_3 handling operations of stages s_1 and s_3. The remaining units are single purpose. From Fig. 12, it follows that products i_1 to i_6 are processed in the first stage s_1 for then being assembled in pairs at stage s_2, generating the products i_7, i_8, and i_9, which are finally processed at stage s_3. Information related to processing times of the products at each processing stage is given in Table 1.

When the illustrative example is solved using the decomposition algorithm, a solution featuring a makespan of 31 h is reported in 0.98 s of CPU time. This solution is equal to the optimal one found by the exact MILP model after 0.25 CPUs. Although the model size is not large (196 linear constraints, 105 binary variables, and 26 continuous variables), the example serves to demonstrate the convergence of the algorithm to the optimal solution of the problem. The computational results are summarized in Table 2 while the optimal solution is illustrated in Fig. 13. From Table 2, it follows that both approaches required less than a second of CPU time to find the optimal solution. Fig. 13 depicts the assignment and sequencing of tasks in each processing unit, wherein every task is identified by the #id of product followed by the id of processing stage; for example, task 2–1 refers to the processing of product #2 on stage #1. In addition, the set of tasks associated to a final product i and its subassemblies $i' \in SA_i$ are depicted with the same color.

5.2. Case study 1

The first case study comes from the pharmaceutical industry, in where the majority of the operations performing in their facilities are batch-type. This industrial case, originally tackled by Nicoară (2012), comprises a sequential job-shop production environment involving 20 processing units, which are grouped in 10 processing stages (see Fig. 14). All processing units are single purpose and hence, the scheduling problem is solved without considering redesign options. The plant produces 16 different types of drugs through product-specific recipes. For each type of drug, Table 3 shows the quantity of batches to produce and its production recipe. Just a subset of units of every processing stage is able to perform the operation required by a given product. This compatibility between products and processing units is also given in Table 3. The processing times depend on both the product and the processing unit and can be obtained from the referenced paper. The general structure of production facility is depicted in Fig. 14, which also illustrates the alternative production routes for the different types of drugs.

The original case study involves the processing of 79 batches on several processing stages. As a result, 606 tasks must be defined for their assignment and sequencing on 20 processing units. This results in an extremely large-size model (see Table 4), which is impossible to be efficiently solved through a pure optimization approach. For this reason, three instances of growing size are generated from the original case study. The computational results obtained are summarized in Table 4. When considering the production of 2 and 4 drugs, involving the scheduling of 58 and 190 tasks, respectively, both solution strategies converge to the same solution but the decomposition procedure achieves it in considerably less computational time than the pure MILP model. The solver can demonstrate the optimality of the solution only in the smallest-size instance. For the full-size example, the solution reporting by the algorithm after 1762 s of CPU time outperforms that one found by the mathematical model after 12 h of computational time. Note that the solutions found by the MILP model for the cases of 4 and 16 drugs feature a high integrality GAP, which is reduced at most in 2.5% without improving the objective value when the CPU time limit imposed for the problem resolution is increased from 1 to 12 h.

5.3. Case study 2

In this case, the proposed methodology is applied to an industrial problem of medium-large size. Such a case study relies on an example previously tackled by Roshanaei et al. (2013) that comprises the production of molds for the automotive industry. A mold is composed of 5 different parts: cavity, core, slide, retractor, and clamp-plates. In the problem definition, both the molds and their corresponding parts represent the products to be processed on the manufacturing line. The original case study taken from the literature involves the processing of 20 parts, corresponding to 4 molds, 8 processing stages, and 14 processing units, without considering mold assembly operations. This paper extends such an example by incorporating a last stage (s_9) with 2 dedicated assembly units.

Table 2
Computational results for the illustrative example.

MILP model						Iterative algorithm		
Cont. var.	Bin. var.	Eqs.	Obj. Func.	CPU(s)	Gap%	Initial solution	Best solution	Total CPU (s)
196	105	26	31	0.25	0	31	31	0.98

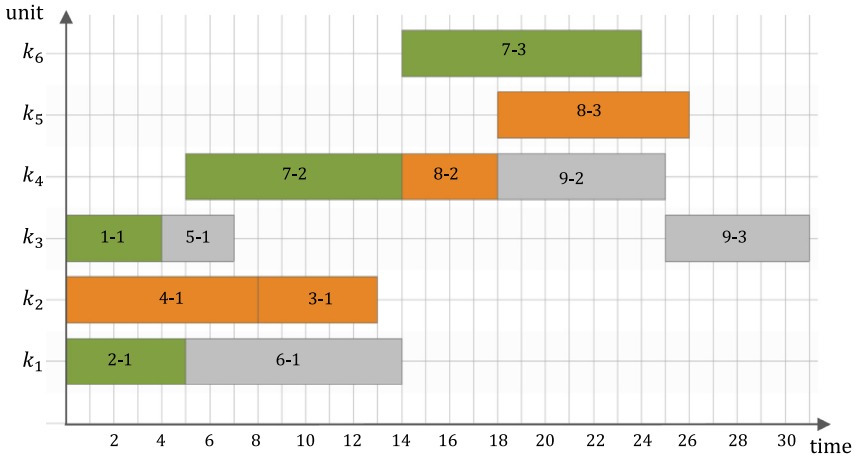


Fig. 13. Optimal schedule for the toy example – $MK = 31$ hours.

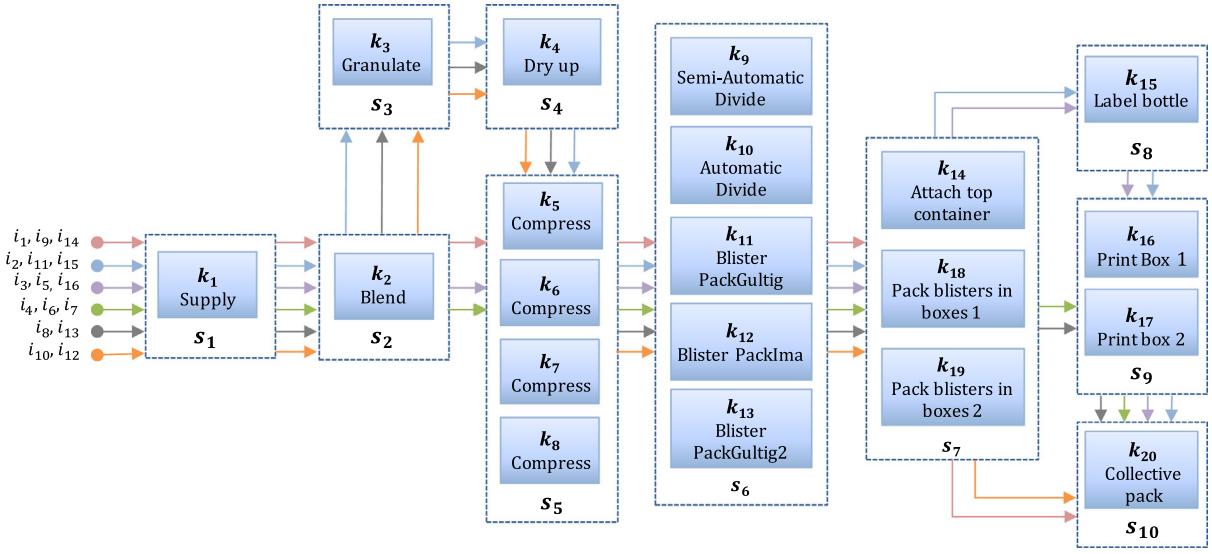


Fig. 14. General structure of the production facility – *case study 1.*

Table 3
Processing sequence for each type of product – *case study 1.*

Product	No. batches	Stages									
		s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
i_1 (Analgin)	8	k_1	k_2			k_7	k_{11}, k_{13}	k_{18}			k_{20}
i_2 (Antacid)	1	k_1	k_2	k_3	k_4	k_8	k_9, k_{10}	k_{14}	k_{15}	k_{17}	k_{20}
i_3 (Ascovit 100)	13	k_1	k_2			k_5, k_6	k_9, k_{10}	k_{14}	k_{15}	k_{17}	k_{20}
i_4 (Ascovit 200)	4	k_1	k_2			k_5, k_7	k_{12}	k_{18}		k_{16}	k_{20}
i_5 (Ascovit 60)	3	k_1	k_2			k_7	k_9, k_{10}	k_{14}	k_{15}	k_{17}	k_{20}
i_6 (Biseptrim cl.)	1	k_1	k_2			k_5, k_6	k_{12}	k_{19}		k_{16}	k_{20}
i_7 (Biseptrim)	19	k_1	k_2			k_5, k_6	k_{12}	k_{18}		k_{16}	k_{20}
i_8 (Ephimigrin)	1	k_1	k_2	k_3	k_4	k_5, k_7	k_{13}	k_{18}		k_{16}	k_{20}
i_9 (Europirin T)	3	k_1	k_2			k_7	k_{11}, k_{13}	k_{18}			k_{20}
i_{10} (Europirin)	3	k_1	k_2	k_3	k_4	k_7	k_{11}, k_{13}	k_{18}			k_{20}
i_{11} (Eurosept)	4	k_1	k_2	k_3	k_4	k_8	k_9, k_{10}	k_{14}	k_{15}	k_{17}	k_{20}
i_{12} (Paracetamol Pus)	2	k_1	k_2	k_3	k_4	k_5	k_{11}	k_{18}			k_{20}
i_{13} (Paracetamol Sinus)	11	k_1	k_2	k_3	k_4	k_5	k_{11}	k_{18}		k_{16}	k_{20}
i_{14} (Paracetamol)	3	k_1	k_2			k_5, k_6	k_{11}	k_{18}			k_{20}
i_{15} (Tussin Forte)	2	k_1	k_2	k_3	k_4	k_5	k_9, k_{10}	k_{14}	k_{15}	k_{17}	k_{20}
i_{16} (Tusin)	1	k_1	k_2			k_5	k_9, k_{10}	k_{14}	k_{15}	k_{17}	k_{20}

Table 4
Computational statistics for the three instances of *case study 1.*

Drugs	Batches	Tasks	MILP model						Iterative algorithm		
			Cont. var.	Bin. var.	Eqs.	Obj. Func.	CPU(s)	Gap%	Initial solution	Best solution	Total CPU (s)
2	9	58	190	283	1429	7171	50.42	–	7171	7171	7.39
4	26	190	595	2423	11,767	10,308	3600*	47.98	10,527	10,308	64.93
						10,308	43,200*	45.51			
16	79	606	1905	21,709	104,903	23,570	3600*	77.02	23,015	22,930	1762
						23,570	43,200*	76.86			

*CPU time limit imposed for the solver execution.

Table 5
Processing sequence for each type of product for *case study 2.*

Type of product	Stages								
	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
Cavity	x	x	x	x	x	x	x	x	
Core	x	x	x	x	x	x	x	x	
Slide	x	x	x	x					
Retractor	x	x							
clamp – plates	x								
mold									x
Units	k_1, k_2,k_3, k_4	k_5	k_3, k_4	k_6, k_7	k_8, k_9 k_{10}	k_{11}	k_3, k_{12}	k_{13}, k_{14}	k_{15}, k_{16}

Table 6
Processing times [hours] for *case study 2.*

Product i	Stages								
	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
i_1- *cavity*	60	85	24	52	40	84	32	26	
i_2- *core*	79	80	24	70	84	84	47	60	
i_3- *slide*	11	35	25	19					
i_4- *retractor*	16	40							
i_5- *clamp – plates*	36								
i_6- *cavity*	62	85	48	52	45	84	21	26	
i_7- *core*	74	74	48	60	65	84	40	60	
i_8- *slide*	30	20	7	20					
i_9- *retractor*	35	36							
i_{10}- *clamp – plates*	45								
i_{11}- *cavity*	60	94	36	52	40	84	21	23	
i_{12}- *core*	72	74	90	60	59	84	46	54	
i_{13}- *slide*	16	20	12	35					
i_{14}- *retractor*	14	16							
i_{15}- *clamp – plates*	45								
i_{16}- *cavity*	55	76	36	54	54	36	32	26	
i_{17}- *core*	67	70	36	60	73	36	26	46	
i_{18}- *slide*	23	21	20	39					
i_{19}- *retractor*	23	38							
i_{20}- *clamp – plates*	50								
i_{21}- *mould*									35
i_{22}- *mould*									30
i_{23}- *mould*									25
i_{24}- *mould*									38

Table 5shows the processing sequence for each type of product and the units performing operations at each processing stage. From this table, it deduces that a total of 96 tasks must be assigned and sequenced in order to find a feasible schedule. Note also that units k_3 and k_4 are multipurpose because k_3 is able to perform operations of stages s_1, s_3, and s_7, while k_4 is suitable for operations of stages s_1 and s_3. Information regarding to processing times of the products at each stage is specified in Table 6.

When this real-life problem is solved by using the exact optimization approach, the optimal solution, reporting a makespan of 979 h, is found after 127.7 s of CPU time, while the decomposition algorithm converges to the same solution in just 28.8 s (see row 1 of Table 7).

In order to test the scalability of the algorithm when increased the number of parts to be processed in the production system, two additional instances considering the production of 6 and 8 molds, respectively, are generated from the original case study. The features of molds 5 to 8 are similar to those ones of molds 1, 2, 3, and 4, respectively. The computational statistics and objective values for each instance considered of case study 2 are summarized in Table 7. This table details the solutions reported by both the constructive and the improvement stage of the iterative algorithm.

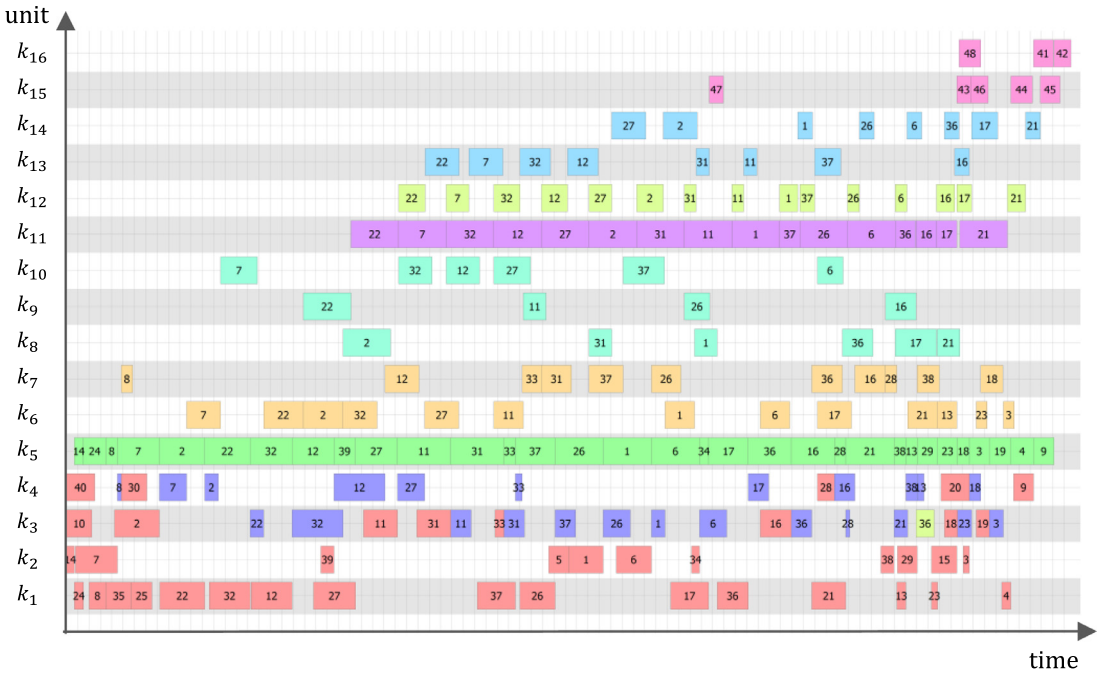
As can be seen in Table 7, the resolution of MILP model considering the full space solution cannot find the optimal makespan for the new problem instances within a time limit of an hour. In the case of 6 molds, the MILP model reports a solution of 1355 h with an integrality gap of 30.5%, while the algorithm reaches the same solution in just 455.7 s. This makes suspect that the solution found is the optimal one but the MIP solver cannot demonstrate it within the time limit predefined. Moreover, when the production of 8 molds is considered, the objective value of 1764 h found by the decomposition procedure in 1145 CPUs time is better than that one achieved by the exact optimization approach ($MK = 1772$ hours) after an hour of CPU time. When the examples involving the production of 6 and 8 molds are solved without setting a maximum CPU time limit, the solver ends due to memory capacity error. In the case of 6 molds, a small reduction of the integrality GAP is achieved after 100 h without improving the objective value while for the example with 8 molds, the same solution found by the algorithm is obtained, but requiring more than 82 h of CPU time. Note that to find a solution for the scheduling problem involving 8 molds, the model faces with the assignment and sequencing of 192 tasks on 16 processing units. In spite of the complexity of the instances considered for *case study 2* (see Table 7), the decomposition approach shows a good computational performance, reporting high-quality solutions with a modest CPU time, for all cases considered.

The best schedules found by each solution strategy for the instance of 8 molds are depicted in Fig. 15. In this picture, the operations performed on each processing unit are colored according to the processing stage to which they belong. For this reason, the multipurpose units (k_3 and k_4) have operations of two different color. Besides, each task is labeled with the #id of the product that is being processed due to the reduced space available in the figure. For instance, mold 1 is the product #48 and its corresponding parts are product #1 to #5. On the other hand, the schedule depicted in Fig. 15a shows as in the multipurpose unit k_3, the prod-

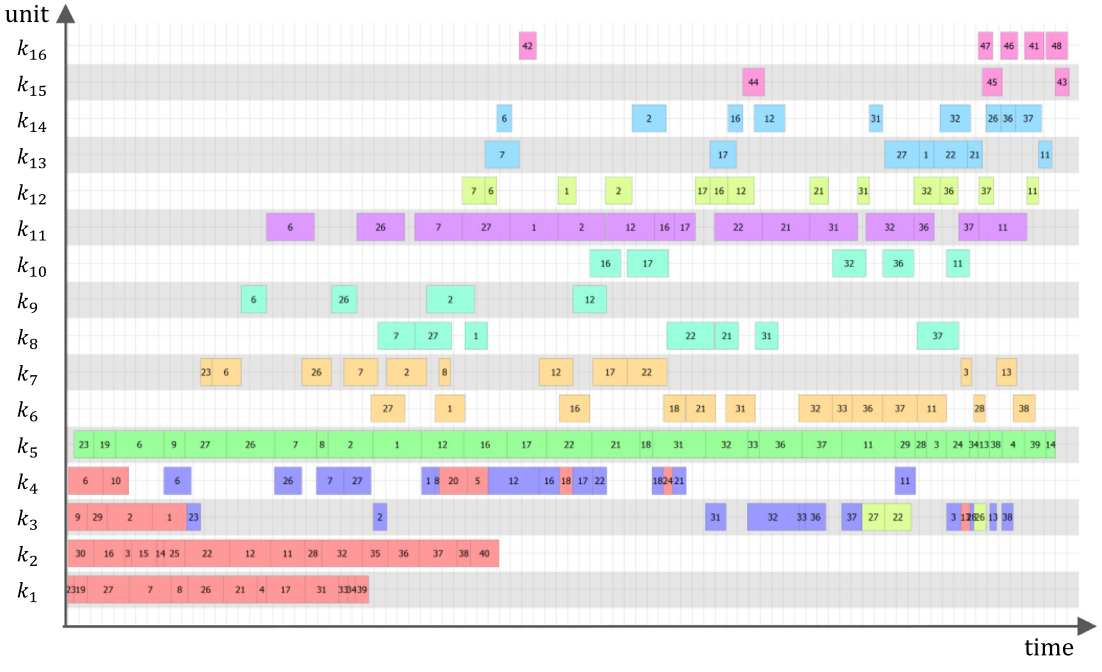
Table 7
Computational statistics for the three instances of *case study 2*.

Molds	MILP model						Iterative algorithm		
	Cont. var.	Bin. var.	Eqs.	Obj. Func.	CPU(s)	Gap%	Initial solution	Best solution	Total CPU (s)
4	194	2332	4685	979	127.7	0	1266	979	28.8
6	290	5166	10,363	1355	3600*	30.5	1633	1355	455.7
				1355	360,038**	27.4			
8	386	9112	18,265	1772	3600*	55.4	2030	1764	1145
				1764	295,233**	46.8			

*CPU time limit imposed for the solver execution.
**Solver ended because the memory capacity was exceeded.



(a) $MK = 1764$ hours



(b) $MK = 1772$ hours

Fig. 15. Scheduling solutions found for *case study 2* with 8 molds: (a) the iterative approach and (b) the MILP model.

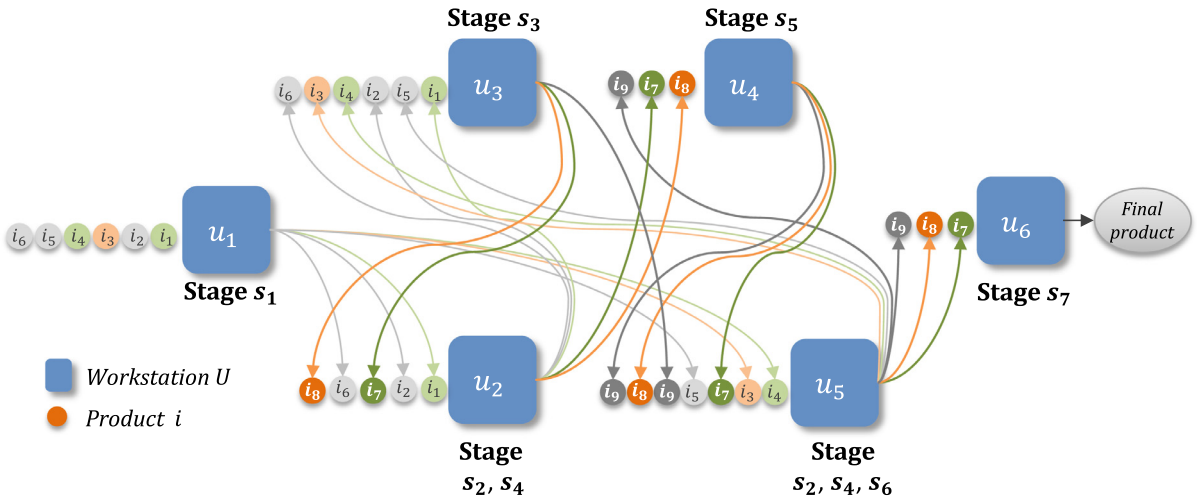


Fig. 16. Processing and assembly stages in shipbuilding process – FJSP-A.

ucts i_{11}, i_{31}, and i_{36} (#11, #31, and #36, respectively) appears twice, representing the processing of products #11 and #31 at stages s_1 and s_3, and the processing of product #36 at stages s_3 and s_7. Finally, it is worth to remark that when the redesign stage is applied to the three instances derived from *case study 2*, no unit could be released in neither of these examples because the scheduling solutions show a high utilization of all processing units.

5.4. Case study 3

This third case study faces with the scheduling and redesign of a multi-stage shipbuilding process arising in the naval industry. The shipbuilding process involves multipurpose units and assembly operations. According to the modular approach to offshore vessel design and configuration, a ship is constructed by assembling a set of blocks, which represent the largest construction units of a ship. Each block in turn is formed from one or more sub-blocks which generally include some welded equipment, such as bushings, pipes, and cables, implementing before painting. Once finishing the production of the blocks, they are transported and positioned in a dry dock for building the ship. An operation known as *Block Erection* is performed to mount the blocks, one after another, according to a predefined sequence given by the specifications of the ship design. The naval company seeks to minimize the total time required to mount a ship due to the high penalty costs incurred by the company when the completion of a project is delayed. Moreover, the inactivity of any resource, such as a dry dock, represents a loss of income for the company.

The real-life example presented here involves the production of a large ship composed by 25 blocks and 50 sub-blocks. The shipyard is integrated by 42 workshops that perform operations of 7 processing stages, wherein 13 workshops are multipurpose: 6 ones handling operations of processing stages s_2 and s_4, while the other 7 ones are shared by stages s_2, s_4, and s_6. There are two processing stages that carry out assembly operations ($|S^a| = 2$), one for assembling the sub-blocks and the other one for mounting the ship. The problem configuration is illustrated in Fig. 16. The products to be processed are represented by circles while the work cells integrating the manufacturing system are depicted by rectangles. As shown in Fig. 16, there are groups of parallel units with single purpose (u_1, u_3, u_4, and u_6) and other ones are multipurpose, such as u_2 (stages s_2 and s_4) and u_5 (stages s_2, s_4, and s_6). The small example depicted in Fig. 16 involves the processing of 6 sub-blocks ($i_1 - i_6$) and 3 blocks (i_7, i_8, and i_9). Sub-blocks are processed at the first two stages and then, they are assembled in pairs at stage s_3 to form the blocks. The remaining stages s_4, s_5, and s_6 perform operations on the blocks, which are put together in stage s_7 to mount the ship. The data about processing times and other problem features, such as the type of operation performing on the sub-blocks/blocks at each processing stage, cannot be given in this paper by confidentiality issues imposed by the company.

The computational results obtained when *case study 3* is solved through both the exact MILP model and the iterative algorithm, are summarized in Table 8. The example results into a huge MILP model of 148,551 equations, 9300 binary variables, and 452 continuous variables. As consequence, the full space approach gives a makespan of 241.7 days, with an integrality gap of 25.2%, after the predefined CPU time limit of an hour. Instead, the decomposition algorithm finds a solution that is 2.8% better (makespan of 235 days) in just 265.2 s of CPU time. Moreover, when a CPU time limit is not imposed for the resolution of the full-space MILP model, the solver ends because the memory capacity is exceeded and reports a solution featuring a makespan of 241.4 days with a GAP value of 25.3%. Once again, the solution of this industrial scale problem illustrates how the strategy of solving iteratively the MILP model with a reduced search space at each solver execution outperforms the full space approach in terms of CPU time and solution quality.

Fig. 17 shows the behavior of the solution found by both approaches over computational time. As can be seen in this picture, the high computational efficiency of the proposed iterative methodology allows to report, in less than 5 min of CPU, an improvement in the solution of 84.7% with regards to the monolithic MILP formulation within the same CPU time. In this case, the algorithm is parameterized with the following values: $N^{max} = 10$ and $lTime = 10$ seconds. Furthermore, when analyzing the performance of the decomposition algorithm by setting different values to these input parameters ($N^{max} = 10$ and 3 different values for the time limit of execution $lTime$, 10, 20, and 30 CPUs), the best solution found evolves as shown in Fig. 18.

From Fig. 18 it follows that, regardless of the value of parameter $lTime$, the makespan is reduced considerably in the first iterations. Then, for values $N = 5$ or greater, there are no improvements in the objective function. The analysis of Fig. 18 allows concluding that the best values for the input parameters are $N^{max} = 5$ and $lTime = 30$ seconds. When the decomposition algo-

Table 8
Computational statistics for *case study 3*.

MILP model			Iterative algorithm			
Objective value	CPU (s)	GAP (%)	Initial solution	CPU (s)	Best solution	CPU*(s)
241.7	3600	25.2	261.2	18.9	235	265.2

*accumulative time including previous algorithmic stages.

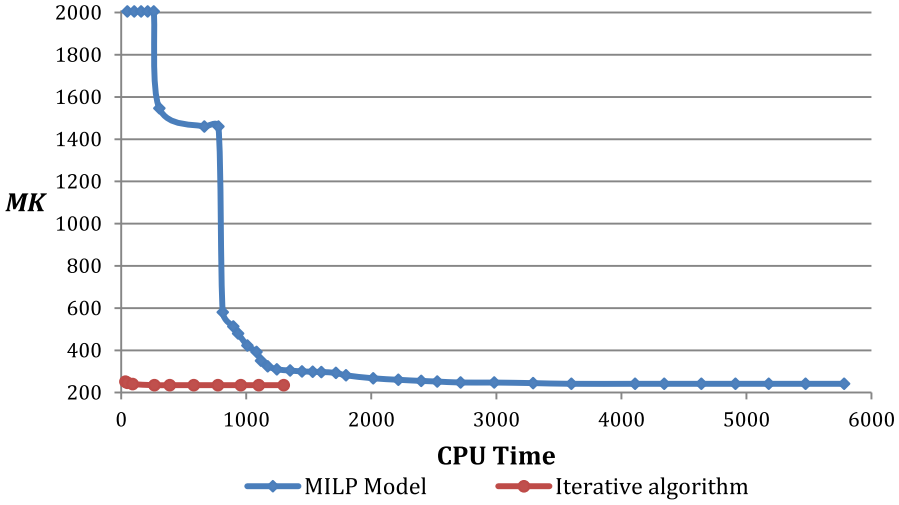


Fig. 17. Evolution of the solutions found by the MILP Model and the decomposition algorithm through time.

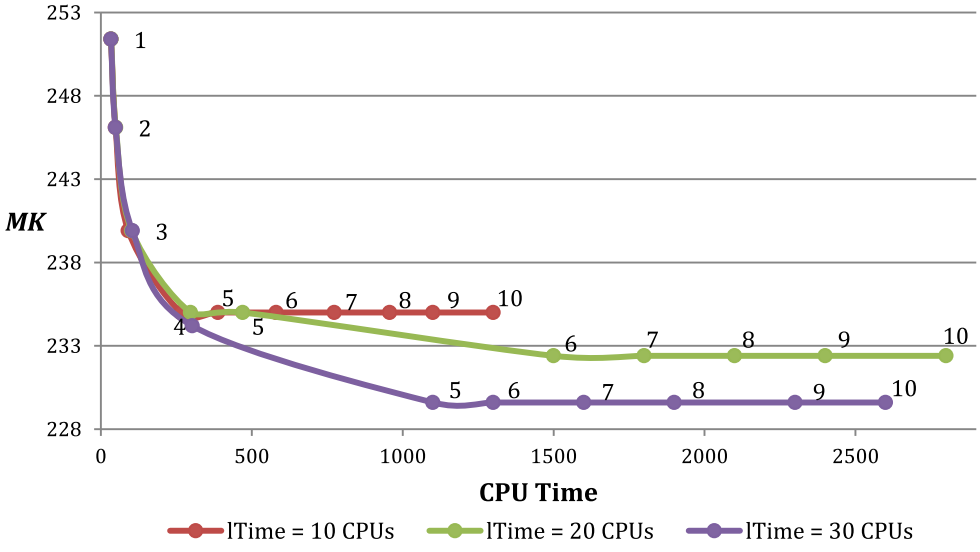


Fig. 18. Solution quality vs. computational time – decomposition algorithm.

rithm is run considering these parameter values, a solution featuring a makespan of 229.6 days is obtained after 1100 s of CPU time. This solution, the best found for *case study 3*, is depicted in Fig. 19 by a Gantt chart. A total of 225 tasks are assigned and sequenced in 42 processing units in order to find a feasible schedule for the example at hand. In this Gantt chart, all tasks belonging to a same processing stage are represented with the same color and each operation is labeled with the #id of the product related to the processing task. Note that several tasks of different colors have been assigned to work cells u_2 and u_5 because the units integrating these workstations are multipurpose. The time currently employed by the company for delivering a ship of size 25 × 50 is about a year and half. Consequently, the new schedule given in Fig. 19 reduces the total time that takes to build a ship by approximately 60%.

Once obtained the best solution for the scheduling problem, the second stage of the iterative procedure is carried out to analyze the redesign of the productive system. At first, the procedure takes as input the scheduling solution given in Fig. 19 and determines that 7 processing units can be released without worsen the makespan of 229.6 days. The units becoming part of set L are: k_{16} and k_{20} of workstation u_2 and k_{22}, k_{23}, k_{26}, k_{29}, and k_{30} of workstation u_3. By analyzing Fig. 19, it is possible to known beforehand that work cell u_3 is oversized. Nevertheless, there is nothing to suggest that the quantity of units in work cell u_2 could be also reduced. Fig. 20 illustrates the new schedule that results after minimizing the num-

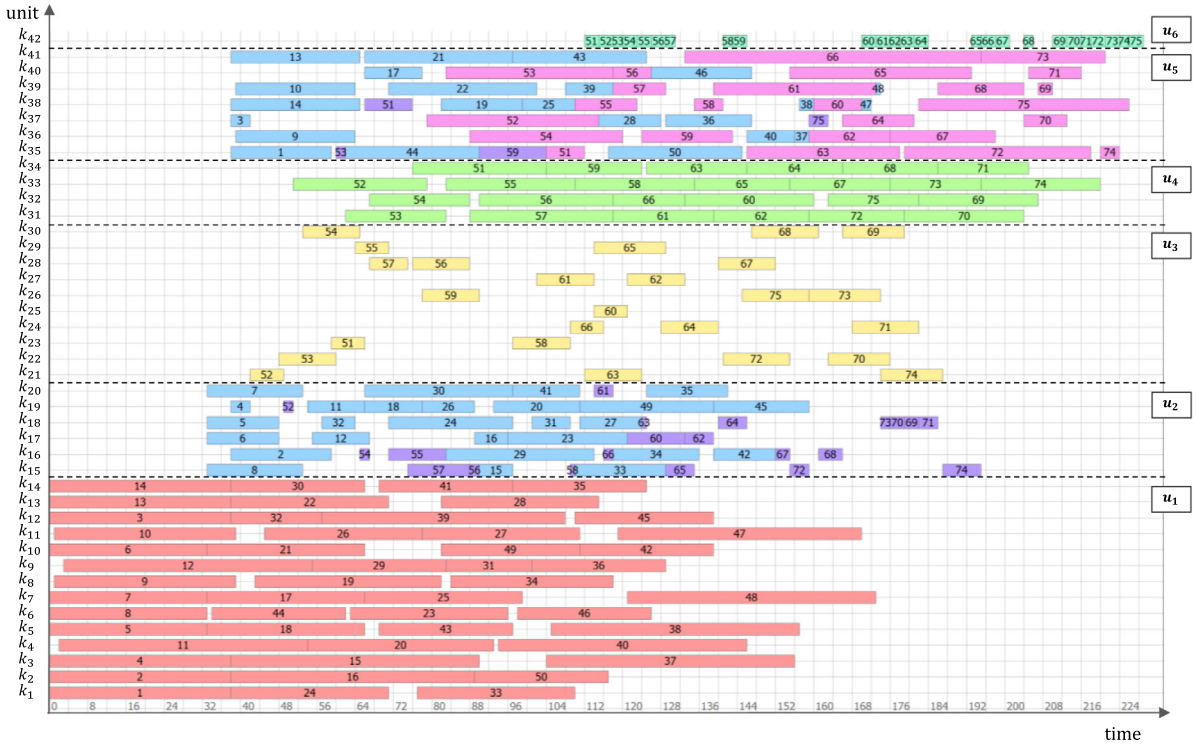


Fig. 19. Best solution found by the iterative algorithm for *case study 3*– $MK = 229.6$ days.

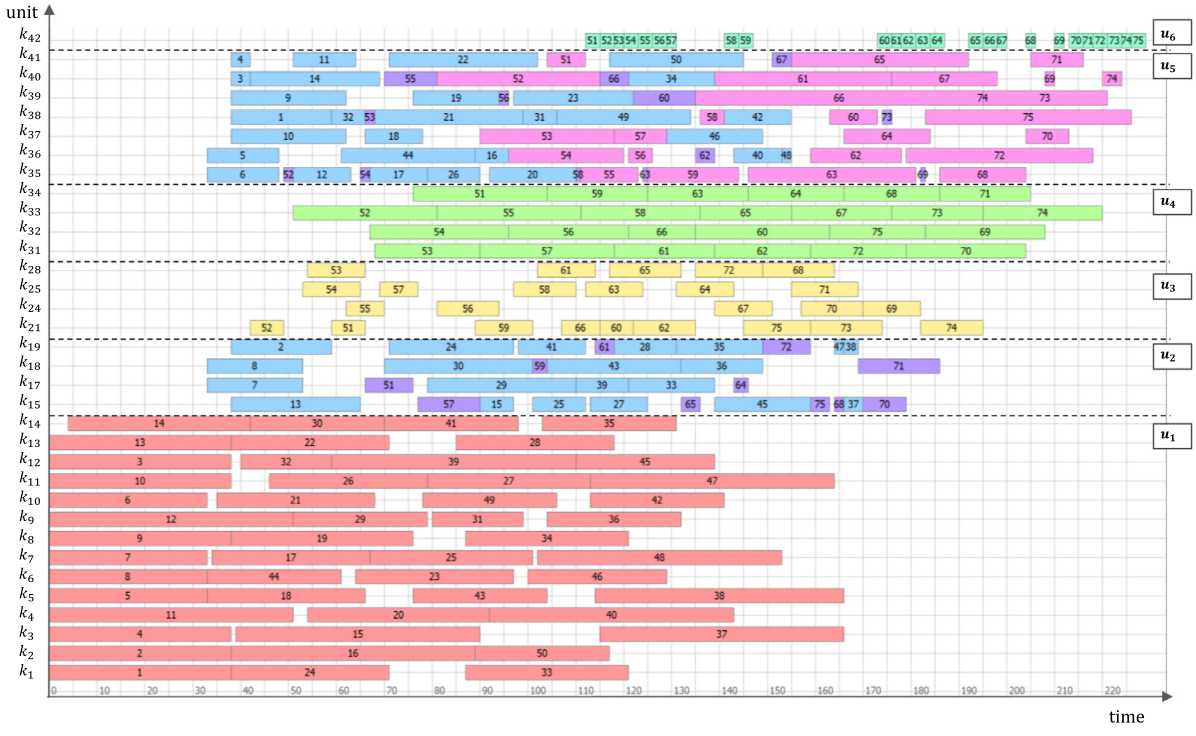


Fig. 20. Best solution for *case study 3* with 7 units released – $MK = 229.6$ days.

ber of units to be utilized at each work cell. Note that tasks originally assigned to released units k_{16}, k_{20}, k_{22}, k_{23}, k_{26}, k_{29}, and k_{30} (see Fig. 19) are reassigned to other units in the schedule depicted in Fig. 20.

Once the set $L = \{k_{16}, k_{20}, k_{22}, k_{23}, k_{26}, k_{29}, k_{30}\}$ is defined, the algorithm continues with the relocation of the processing units $k \in L$ to any compatible workstations $u \in U_k$. In this example, as the units represent workshops of the shipyard, any facility can be suitable to perform operations of any processing stage, except the painting operations, which should be accomplished in workshops arranged for such a purpose.

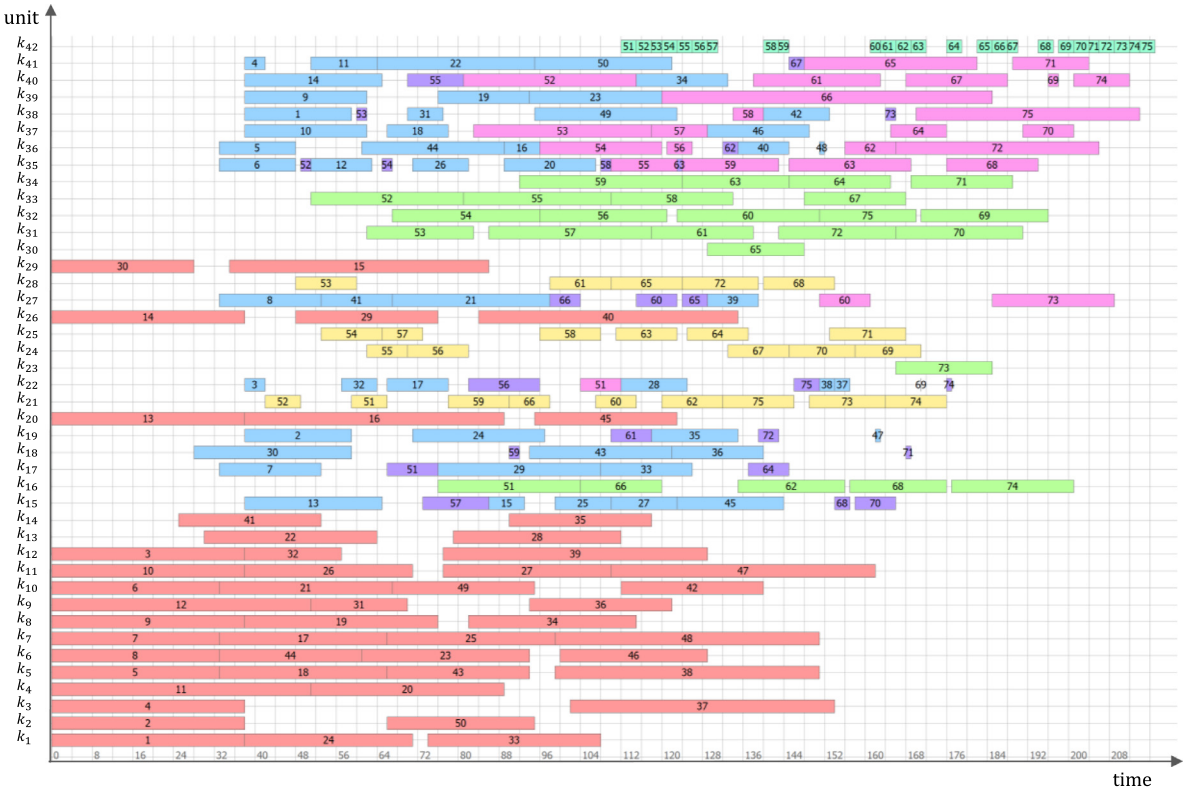


Fig. 21. Best solution reported by the *alternative 1* for *case study 3* – $MK = 217.5$ days.

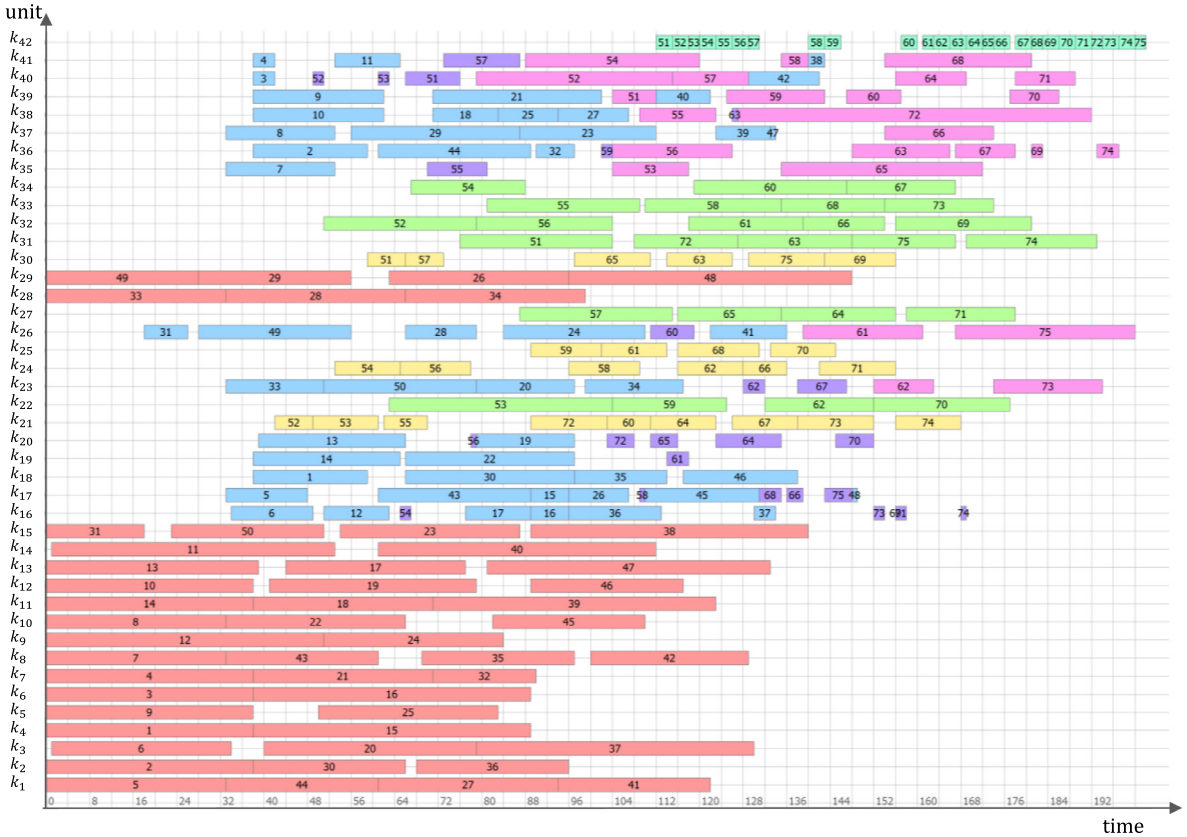


Fig. 22. Best solution reported by the *alternative 3* for *case study 3* – $MK = 202.9$ days.

Table 9
Computational statistics of the integrated problem of scheduling and redesign for *case study 3*.

Heuristic	Scheduling stage				Redesign stage			
	Initial solution	CPU (s)	Best solution	CPU* (s)	Released units	CPU* (s)	Redesign solution	CPU* (s)
1	261.2	19.0	229.6	1139.3	7	12,337.0	217.5	12,497.7
2	261.2	18.1	229.6	1147.0	7	12,355.7	216.3	15,957.7
3**	261.2	17.8	229.6	1141.7	7	12,353.8	202.9	188,967.6

*accumulative time including previous algorithmic stages.
**unlimited runtime in the last phase of step redesign, the solver ends up exceeding the memory capacity.

The problem of relocating the released units is solved by using both the iterative strategy proposed by the algorithm (*alternative 1*) and the pure MILP model (*alternative 2 and 3*). The computational results obtained are summarized in Table 9. In *alternative 1*, the units are relocated iteratively, one by one. As a result, the algorithm converges to a solution featuring a makespan of 216.3 days after 160 s of CPU time. On the other hand, *alternative 2* and *3* solve the full space MILP modelconsidering all units $k \in L$ in a simultaneously way. In *alternative 2*, a CPU time limit of 3600 s is imposed for the solver execution, while such a termination criterion is neglected in *alternative 3*. From Table 9, it follows that *alternative 3* finds the best solution (202.9 days) but employing almost 50 h of CPU time. In this case, the solver ends because the memory capacity was exceeded.

Taking into consideration that the redesign problem involves long-term decisions, the high computational expense exhibited by *alternative 3* represents a minor issue in the problem resolution. The best scheduling solution found after plant reconfiguration for alternative 1 and 3 are depicted in Figs. 21 and 22, respectively. In the last case, the makespan is reduced around 12%, from 229.6 to 202.9 days, with regards to the original plant configuration.

6. Conclusions

This paper has addressed the integrated scheduling and redesign problem of flexible manufacturing plants involving multipurpose units and assembly operations. A rigorous mathematical model relied on the general precedence concept was developed for solving the problem under study. The MILP model incorporates the definition of task in order to handle the allocation and sequencing decisions in multipurpose units. Furthermore, the paper proposes a set of novel constraints for facing with the redesign problem. The main goal is the minimization of the makespan.

Even though the full space approach is able to find the optimal solution for small problem instances, its applicability to industrial size problems is either impossible or yield poor solutions. Therefore, the MILP model is embedded within a decomposition algorithm, which solves repeatedly the rigorous model but considering a reduced search space at each solver execution. The iterative procedure is integrated by two main stages: one for solving the scheduling problem and the other one for the redesign problem. In the scheduling stage, the products to be processed are incorporated one by one together with their subassemblies until a complete feasible initial schedule is obtained. Then, this initial solution is improved iteratively by executing several rescheduling actions. Once the best schedule for the current plant configuration is determined by the procedure, this solution is taking as input for the redesign stage. The first goal of redesign stage is to determine if the plant is oversized by minimizing the number of units used in each workstation, and then, in a second objective, the redesign problem evaluates alternative plant reconfigurations in order to minimize the makespan found at first by the scheduling stage.

The proposed algorithm was used for solving several instances derived from three real-life case studies. In order to determine the complexity of the examples selected and the quality of the solutions reported by the decomposition method, all problem instances were also solved by using the full space MILP model. Computational results demonstrate that the algorithm converges efficiently to the optimal solution with low computational effort when small-to medium sized instances are considered. For the case of large-scale problems, the iterative procedure reports better solutions than the rigorous mathematical model, also outperforming this optimization approach in terms of CPU time.

Finally, it is worth to remark that the iterative procedure can be easily adapted for solving other types of scheduling problems by changing appropriately the mathematical model used as base.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Natalia P. Basán: Software, Validation, Methodology, Investigation, Writing - review & editing. **Mariana E. Cóccola:** Software, Validation, Methodology, Investigation, Writing - review & editing. **Alejandro García del Valle:** Supervision. **Carlos A. Méndez:** Conceptualization, Investigation, Writing - review & editing, Project administration, Funding acquisition, Supervision.

Acknowledgments

The authors gratefully acknowledge the financial support from CONICET under Grant PIP 112 20150100641 and from Universidad Nacional del Litoral under Grant CAI+D 2016-UNL / PIC 50420150100101LI.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.compchemeng.2020.106777.

References

Aguirre, A.M., Méndez, C.A., Gutierrez, G., De Prada, C., 2012. An improvement-based MILP optimization approach to complex AWS scheduling. Comput. Chem. Eng. 47, 217–226. doi:10.1016/j.compchemeng.2012.06.036.

Aguirre, A.M., Papageorgiou, L.G., 2018. Medium-term optimization-based approach for the integration of production planning, scheduling and maintenance. Comput. Chem. Eng. 0, 1–21. doi:10.1016/j.compchemeng.2018.04.030.

Basan, N.P., Achkar, V.G., Mendez, C.A., Garcia-Del-Valle, A., 2018. A heuristic simulation-based framework to improve the scheduling of blocks assembly and the production process in shipbuilding, in: proceedings - Winter Simulation Conference. 10.1109/WSC.2017.8248040.

Basán, N.P., Achkar, V.G., Méndez, C.A., García-del-Valle, A., 2017. A hybrid simulation-based optimization approach for scheduling. In: 29th European Modeling and Simulation Symposium, EMSS 2017, Held at the International Multidisciplinary Modeling and Simulation Multiconference, I3M 2017, pp. 83–90.

Basán, N.P., Cóccola, M.E., García del Valle, A., Méndez, C.A., 2019 a. An efficient MILP-based decomposition strategy for solving large-scale scheduling problems in the shipbuilding industry. Optim. Eng. 20, 1085–1115. doi:10.1007/s11081-019-09457-y.

Basán, N.P., Cóccola, M.E., García del Valle, A., Méndez, C.A., 2019 b. An efficient MILP-Based decomposition strategy for solving large-scale scheduling problems in the offshore oil and gas industry, in: computer aided chemical engineering. pp. 943–948. 10.1016/B978-0-12-818634-3.50158-2.

Basán, N.P., Cóccola, M.E., García del Valle, A., Méndez, C.A., 2019 c. An effective MILP-based decomposition algorithm for the scheduling and redesign of flexible job-shop plants. Chem. Eng. Trans. 74, 613–618. doi:10.3303/CET1974103.

Basán, N.P., Cóccola, M.E., Méndez, C.A., 2015. Conducting experimental design and optimization on an innovative car rental business. In: 27th Eur. Model. Simul. Symp. EMSS 2015, pp. 166–171.

Basán, N.P., Méndez, C.A., 2016. Hybrid MILP/Simulation/Heuristic algorithms to complex hoist scheduling problems. Compu. Aided Chem. Eng. doi:10.1016/B978-0-444-63428-3.50326-X.

Blazewicz, J., Dror, M., Weglarz, J., 1991. Mathematical programming formulations for machine scheduling: a survey. Eur. J. Oper. Res. 51, 283–300. doi:10.1016/0377-2217(91)90304-E.

Castro, P.M., Aguirre, A.M., Zeballos, L.J., Méndez, C.A., 2011. Hybrid mathematical programming discrete-event simulation approach for large-scale scheduling problems. Ind. Eng. Chem. Res. 50, 10665–10680. doi:10.1021/ie200841a.

Castro, P.M., Barbosa-Póvoa, A.P., Novais, A.Q., 2005. A design and scheduling RTN continuous-time formulation. Comput. Aided Chem. Eng. 20, 1213–1218. doi:10.1016/S1570-7946(05)80044-6.

Castro, P.M., Grossmann, I.E., Zhang, Q., 2018. Expanding scope and computational challenges in process scheduling. Comput. Chem. Eng. 0, 1–28. doi:10.1016/j.compchemeng.2018.01.020.

Cebral-Fernandez, M., Crespo-Pereira, D., Garcia-Del-Valle, A., Rouco-Couzo, M., 2016. Improving planning and resource utilization of a shipbuilding process based on simulation. In: 28th Eur. Model. Simul. Symp. EMSS 2016, pp. 197–203.

Cerdá, J., Henning, G.P., Grossmann, I.E., 1997. A mixed-integer linear programming model for short-term scheduling of single-stage multiproduct batch plants with parallel lines. Ind. Eng. Chem. Res. 36, 1695–1707. doi:10.1021/ie9605490.

Chibeles-Martins, N., Pinto-Varela, T., Barbósa-Póvoa, A.P., Novais, A.Q., 2010. A Meta-Heuristics Approach For the Design and Scheduling of Multipurpose Batch plants, Computer Aided Chemical Engineering. Elsevier B.V. doi:10.1016/S1570-7946(10)28220-2.

Corsano, G., Montagna, J.M., Iribarren, O.A., Aguirre, P.A., 2007. Heuristic method for the optimal synthesis and design of batch plants considering mixed product campaigns. Ind. Eng. Chem. Res. 46, 2769–2780. doi:10.1021/ie0608049.

Dauzère-Pérès, S., Paulli, J., 1997. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Ann. Oper. Res. 70, 281–306. doi:10.1023/A:1018930406487.

Demir, Y., Kürşat Işleyen, S., 2013. Evaluation of mathematical models for flexible job-shop scheduling problems. Appl. Math. Model. 37, 977–988. doi:10.1016/j.apm.2012.03.020.

Fattahi, P., Jolai, F., Arkat, J., 2009. Flexible job shop scheduling with overlapping in operations. Appl. Math. Model. 33, 3076–3087. doi:10.1016/j.apm.2008.10.029.

Fattahi, P., Saidi Mehrabad, M., Jolai, F., 2007. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. J. Intell. Manuf. 18, 331–342. doi:10.1007/s10845-007-0026-8.

Fumero, Y., Corsano, G., Montagna, J.M., 2012. A mixed integer linear programming model for simultaneous design and scheduling of flowshop plants. Appl. Math. Model. 37, 1652–1664. doi:10.1016/j.apm.2012.04.043.

Garey, M.R., Johnson, D.S., Sethi, R., 1976. The complexity of flowshop and jobshop scheduling. Math. Oper. Res. 1, 117–129. doi:10.1287/moor.1.2.117.

Georgiadis, G.P., Elekidis, A.P., Georgiadis, M.C., 2019. Optimization-Based scheduling for the process industries : from theory to real-life. Processes 7, 438. doi:10.3390/pr7070438.

Harjunkoski, I., Bauer, R., 2017. Industrial scheduling solution based on flexible heuristics. Comput. Chem. Eng. 106, 883–891. doi:10.1016/j.compchemeng.2017.02.018.

Harjunkoski, I., Maravelias, C.T., Bongers, P., Castro, P.M., Engell, S., Grossmann, I.E., Hooker, J., Méndez, C., Sand, G., Wassick, J., 2014. Scope for industrial applications of production scheduling models and solution methods. Comput. Chem. Eng. 62, 161–193. doi:10.1016/j.compchemeng.2013.12.001.

Kondili, E., Pantelides, C.C., Sargent, R.W.H., 1993. A general algorithm for short-term scheduling of batch operations—I. MILP formulation. Comput. Chem. Eng. 17, 211–227. doi:10.1016/0098-1354(93)80015-F.

Kopanos, G.M., Méndez, C.A., Puigjaner, L., 2010. MIP-based decomposition strategies for large-scale scheduling problems in multiproduct multistage batch plants: a benchmark scheduling problem of the pharmaceutical industry. Eur. J. Oper. Res. 207, 644–655. doi:10.1016/j.ejor.2010.06.002.

Kopanos, G.M., Puigjaner, L., Maravelias, C.T., 2011. Production planning and scheduling of parallel continuous processes with product families. Ind. Eng. Chem. Res. 50, 1369–1378. doi:10.1021/ie100790t.

Lee, H., Maravelias, C.T., 2017a. Mixed-integer programming models for simultaneous batching and scheduling in multipurpose batch plants. Comput. Chem. Eng. 106, 621–644. doi:10.1016/j.compchemeng.2017.07.007.

Lee, H., Maravelias, C.T., 2017b. Discrete-time mixed-integer programming models for short-term scheduling in multipurpose environments. Comput. Chem. Eng. 107, 171–183. doi:10.1016/j.compchemeng.2017.06.013.

Li, Z., Ierapetritou, M., 2008. Process scheduling under uncertainty: review and challenges. Comput. Chem. Eng. 32, 715–727. doi:10.1016/j.compchemeng.2007.03.001.

Lin, X., Floudas, C.A., 2001. Design, synthesis and scheduling of multipurpose batch plants via an effective continuous-time formulation. Comput. Chem. Eng. 25, 665–674. doi:10.1016/S0098-1354(01)00663-9.

Liu, S., Pinto, J.M., Papageorgiou, L.G., 2010. Single-Stage scheduling of multiproduct batch plants : an edible-oil deodorizer case study. Ind. Eng. Chem. Res. 49, 8657–8669. doi:10.1021/ie1002137.

Maravelias, C.T., 2006. A decomposition framework for the scheduling of single- and multi-stage processes. Comput. Chem. Eng. 30, 407–420. doi:10.1016/j.compchemeng.2005.09.011.

Maravelias, C.T., 2012. General framework and modeling approach classification for chemical production scheduling. AIChe J 58, 1812–1828. doi:10.1002/aic.13801.

Méndez, C., Henning, G., Cerdá, J., 2000. Optimal scheduling of batch plants satisfying multiple product orders with different due-dates. Comput. Chem. Eng. 24, 2223–2245. doi:10.1016/S0098-1354(00)00584-6.

Méndez, C.A., Cerdá, J., Grossmann, I.E., Harjunkoski, I., Fahl, M., 2006. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Comput. Chem. Eng. 30, 913–946. doi:10.1016/j.compchemeng.2006.02.008.

Méndez, C.A., Henning, G.P., Cerdá, J., 2001. An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. Comput. Chem. Eng. 25, 701–711. doi:10.1016/S0098-1354(01)00671-8.

Nicoară, E.S., 2012. Multi-Objective flexible job shop scheduling optimization models (II). Econ. Insights Trends Chall. LXIV, 79–86.

Pan, C.-.H., 1997. A study of integer programming formulations for scheduling problems. Int. J. Syst. Sci. 28, 33–41. doi:10.1080/00207729708929360.

Pinedo, M.L., 2016. Scheduling: theory, algorithms, and systems. Scheduling: Theory, Algorithms, and Systems, 5th Edit. ed Springer, New York doi:10.1007/978-3-319-26580-3.

Pinto, J.M., Grossmann, I.E., 1995. A continuous time mixed integer linear programming model for short term scheduling of multistage batch plants. Ind. Eng. Chem. Res. 34, 3037–3051. doi:10.1021/ie00048a015.

Pranzo, M., Meloni, C., Pacciarelli, D., Bari, P., Orabona, E., 2003. A new class of greedy heuristics for job shop scheduling problems. Lect. Notes Comput. Sci. 2647, 223–236. doi:10.1007/3-540-44867-5_19.

Roshanaei, V., Azab, A., Elmaraghy, H., 2013. Mathematical modelling and a meta-heuristic for flexible job shop scheduling. Int. J. Prod. Res. 51, 6247–6274. doi:10.1080/00207543.2013.827806.

Rossi, A., 2014. Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships. Int. J. Prod. Econ. 153, 253–267. doi:10.1016/j.ijpe.2014.03.006.

Shen, L., Dauzère-Pérès, S., Neufeld, J.S., 2018. Solving the flexible job shop scheduling problem with sequence-dependent setup times. Eur. J. Oper. Res. 265, 503–516. doi:10.1016/j.ejor.2017.08.021.

Verbiest, F., Cornelissens, T., Springael, J., 2019 a. A matheuristic approach for the design of multiproduct batch plants with parallel production lines. Eur. J. Oper. Res. 273, 933–947. doi:10.1016/j.ejor.2018.09.012.

Verbiest, F., Pinto-Varela, T., Cornelissens, T., Springael, J., Barbosa-Povoa, A., 2019 b. Decomposition approaches for the design and scheduling of multiproduct multistage batch plants with parallel lines. Comput. Chem. Eng. 127, 111–126. doi:10.1016/j.compchemeng.2019.05.001.

Xia, W., Wu, Z., 2005. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Comput. Ind. Eng. 48, 409–425. doi:10.1016/j.cie.2005.01.018.