Optimization of the Scheduling of the Compound Production Machines in a Tires Factory

Carlos Gorria and Mikel Lezaun

Abstract The target of the investigation is to efficiently organize the scheduling of the rubber components manufacturing of a vehicle tire production plant. A mathematical formulation followed by a computational code have to be found in order to optimize the assignment of the tasks to the machines. The complexity of the problem lies in the large number of different components to be considered together with the limitations in the compatibility between some machines and components. In addition, the production flow depends on several sequentially ordered sets of products, that comprise from grinding the raw material until manufacturing the final product ready to be assembled. Occasionally, urgent incoming demand of products can cause a sudden change in the factory environment that needs a fast answer. In this scenario, operations research tools and optimization models become crucial for calculating at any given moment a feasible solution that reaches the new constraints. A linear discrete formulation is suitable to deal with the problem. The results of the simulations clearly improving manufacturing productivity and competitiveness.

1 Introduction

A tire factory is an infrastructure where complex processes take place. In this environment, the optimization techniques can be very useful when efficiently planning of sequences of jobs or task chains has to be planned. Many examples of investigations about process optimization in companies in the sector of tire manufacturing can be found, for example in references [10, 12] and [21].

A car tire is built of several different rubber layers. Each layer will provide different property to the tire, indeed, resistance, flexibility or adhesion. Each one of these layers is the product of the processing of a sequence of tasks. Each task consists in the mixture of a rubber compound with a particular pigment in big

C. Gorria (✉) · M. Lezaun
Department of Applied Mathematics, Statistics and Operations Research, University of the Basque Country (UPV/EHU), Leioa, Spain
e-mail: carlos.gorria@ehu.eus; mikel.lezaun@ehu.eus

M. Cruz et al. (eds.), *Progress in Industrial Mathematics: Success Stories*,
SEMA SIMAI Springer Series 5, https://doi.org/10.1007/978-3-030-61844-5_7

rubber mixing machine to become a new compound. It means that before starting the processing of a compound, certain quantity of the preceding one has to be available in stock. The last compound of a sequence is called "final stage compound". Strips of determinate size of each of the final compounds are cut in order to assemble progressively the tire.

In the processing of a sequence of tasks each compound is associated with a different stage that requires a particular manufacturing process. Depending on the design of a machine in the factory, it may be multi-functional having the capability for processing a variety of compounds. The problem of the optimal assignment of tasks to machines was analyzed in references [6, 20] and [26], where conditions imposed by the skills were considered. The work-flow process considering the interactions between compounds and compatible machines can be represented by a directed graph. There the compounds are the nodes and the machines available to process a compound represent the directed edges or arcs of the graph. Several examples of the use of graphs to schematize the flow of an industrial process can be found in the literature [1, 22] and [3].

The sequences of products or task chains considered for the scheduling can be complete, or incomplete as a result of inventory replenishment operations or due to rearranging previously aborted productions. The graph of Fig. 1 shows several product chains, one complete, $\{R_1, O_{12}, \ldots, O_{15}\}$, another incomplete, $\{O_{43}, \ldots, O_{45}\}$, and two chains sharing the same raw material, $\{R_2, O_{22}, \ldots, O_{24}\}$, and, $\{R_2, O_{32}, \ldots, O_{35}\}$.

Eventually, one or more compounds can compete for a common raw material, but we will consider this option here only in the second stage. It means that different chains could share the same original raw material obtained from the initial grind process. This situation has been illustrated in Fig. 1, where the original compound R_2 is the common raw material for compounds O_{22} and O_{32}. For example, before

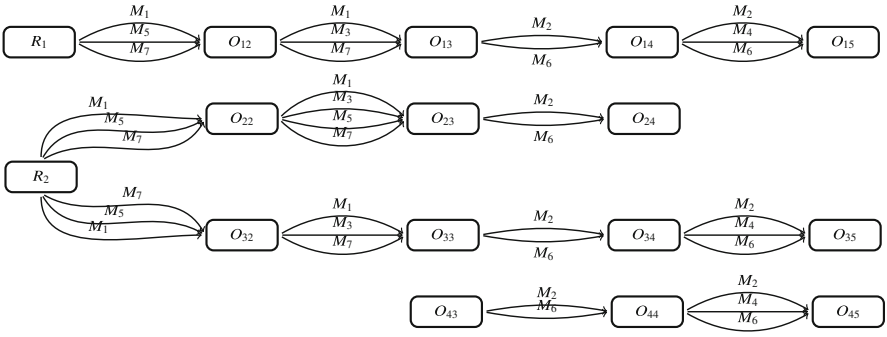

Fig. 1 The graph represents an example of the process flow for some sequences of rubber compounds, represented by nodes, ranging from the initial raw material R_i to the rest of intermediate stages O_{ij} as far as the final compound O_{il_i}. Several machines M_k, represented by arcs, are available to process a compound O_{ij} by mixing the previous one O_{ij-1} with the corresponding pigments

allowing the production of compound O_{22} at determinate moment the scheduling has to take into account the stock of compound R_2, the previous production of the this compound and consumption made by the competitors.

The aim of this work is to build a model suitable to obtain efficient scheduling for the job processing in the mixing machines, minimizing in some sense the idle time of the machines. This challenge can be categorized in the classical *job shop problem*, introduced initially in Ref. [19] and widely analyzed in the literature, [5, 8, 9, 14, 24] and [28]. In general, the problem of finding optimal scheduling for manufacturing in different industrial sectors has been studied in detail in the last decades by several authors, see [15, 16] and [13]. The analysis of the computational performance of the different formulations and algorithms designed to deal with the job shop problem has been a source for several papers as [2] and [11].

In the problems about optimal scheduling different criteria for optimization can be established. One of the most common is to minimize the makespan, that measures the time interval between starting and ending the whole set of jobs. This measurement can be calculated by each machine or on a global scale. In the first case, the optimization is equivalent to the minimization of the sum of gaps or the idle time of the machines. Other criteria can make us of a penalty weights proportional to the tardiness related to the deadline compliance, see, for example Ref. [31].

In this work we use an linear integer programming formulation together with a branch and bound algorithm. This formulation have been used previously in several papers as [7] and [25].

Traditionally it has been observed a computational impracticability of finding global optima or even feasible solutions of many job shop scheduling for large scale problems. Unfortunately, in practice, this case is very usual. With the aim of going beyond this obstacle, recently, heuristic techniques based on genetic algorithms and tabu search have been developed to find computationally efficient feasible solutions in this field. Some hints in this direction can be found in the papers [4, 18, 27–29] and [26].

This investigation has been developed in the frame of a collaboration with a large tire manufacturing company. The aim of the project was to design an automatic and fast method to calculate feasible scheduling for the machines repeatedly. In practice a scheduling will cover one working day, but because of frequent incidents, maintenance service or incoming urgent orders, during the day successive run of the method have to be made. In that circumstances a quick response proposing feasible scheduling that considers the previous incomplete manufacturing is of fundamental importance.

In Sect. 2 we will describe the manufacturing circumstances that determine the parameters of the model. In Sect. 3 we will present the mathematical formulation related to this particular job shop problem, the variables, the constraints and the objective function. In Sect. 4 a simulation will be shown. Some possible applications and conclusions are discussed in the last section.

2 The Job Shop Problem Conditioned by the Environment

Here we present a job shop environment with a catalog of $i = 1, \ldots, I$ jobs. Each job is and ordered set of tasks, $O_i = \{O_{ij}\}_{j=1}^{l_i}$, where l_i is the length of the ith job or task chain. A task consists on processing a compound O_{ij} together with certain pigment giving rise to the next compound of the job, O_{ij+1}. Usually the job i has a deadline p_i which would not have to be exceeded.

We consider $\{M_k\}_{k=1}^{K}$ machines available to process the jobs. The time interval when the machine k is active is $[S_k, E_k]$, which is included in the global time frame $[1, T]$ of the simulation. Each machine k is only compatible with a set of compounds, defined as $\boldsymbol{\Phi_k}$. Equivalently, the compound O_{ij} can be processed only by set of machines $\boldsymbol{\Lambda_{ij}}$. When $O_{ij} \in \boldsymbol{\Phi_k}$ we shall say that the machine M_k and the compound O_{ij} are compatible. An example about the compatibility between machines and compounds is shown in Table 1. There, the suitability of a machine k for processing a compound O_{ij} is expressed by 1 and the unsuitability by 0. The information expressed in this table is equivalent to the graph of Fig. 1.

The basic ideas of the scheme is that a machine cannot process more than one task at the same time and the flow conditions have to be guaranteed every time. The second condition means that a task O_{ij} cannot start until enough quantity of previous compound O_{ij-1} is available in the system. The inventory of the compound O_{ij-1} at any time depends on the initial stock plus the quantity produced earlier than the order for processing O_{ij} minus the quantity consumed by the competitors of O_{ij} by the same raw material O_{ij-1}.

In the model we will consider the possibility of overlapping in time the manufacturing of two consecutive compounds belonging to the same job, O_{ij-1} and O_{ij}, when different machines are assigned to each of them. The processing of

Table 1 Matrix of suitability (1) or unsuitability (0) for a machine M_k to produce compound O_{ij}

	M_1	M_2	M_3	M_4	M_5	M_6	M_7
O_{12}	1	0	0	0	1	0	1
O_{13}	1	0	1	0	0	0	1
O_{14}	0	1	0	0	0	1	0
O_{15}	0	1	0	1	0	1	0
O_{22}	1	0	0	0	1	0	1
O_{23}	1	0	1	0	1	0	1
O_{24}	0	1	0	0	0	1	0
O_{32}	1	0	0	0	1	0	1
O_{32}	1	0	0	0	1	0	1
O_{33}	1	0	1	0	0	0	1
O_{34}	0	1	0	0	0	1	0
O_{35}	0	1	0	1	0	1	0
O_{44}	0	1	0	0	0	1	0
O_{45}	0	1	0	1	0	1	0

O_{ij} may start as soon as the 30% of the previous compound O_{ij-1} has been already made.

The time length taken by the machine M_k to process the task O_{ij} is measured by an integer number of time units $d_{ijk} \in \mathbb{N}$, according with the discretization of the time frame. The components d_{ijk} are stored in the array D.

The manufacturing of certain compounds O_{ij} in a machine M_k may produce some impurity which could cause damages in certain compounds $O_{i'j'}$ processed immediately after in the same machine M_k. In those cases it would be convenient to proceed with a brief cleaning process of $h_{iji'j'}$ time length, in between both manufacturing, in order to avoid eventually damages.

The use of products in stock and the possibility of ordering jobs for inventory replenishment purposes are considered in the mathematical model. The stock of compound O_{ij} is denoted as S_{ij}. The compounds eventually available in stock can be introduced in the task flow. In the same way, when the inventory of a very demanded compound O_{ij} is below a security level, then the incomplete task chain $\{O_{i1}, \ldots, O_{ij}\}$ which leads to this critical compound can be ordered. This type of sequences dedicated to inventory replenishment often are scheduled in order to fill the idle time of the machines. When an interruption of the machines causes the stop of the processing of some jobs, the compounds manufactured up to that moment become part of the stock.

Let define FJ_{ij} the end time of the O_{ij}. Therefore, the end time of the job i will be $FJ_i = FJ_{il_i}$. At the same time, let define FM_k the end time of the last task in the scheduling assigned to machine M_k. Then, since the time is discretized in this mathematical model, M_k machine will be free to process new tasks from $FM_k + 1$ moment. It means that a new scheduling could be developed for a new set of jobs taking $S_k = FM_k + 1$ as he new activation time of any M_k machine. The new scheduling may join the previous one in order to have a long scheduling. We can't guarantee the global optimality of this long scheduling, but it might be very practical in order to evaluate the possibility of manufacturing the jobs in time.

3 Mathematical Formulation of the Problem

In the last decades several algorithms have been developed for attacking a wide variety of problems encompassed under the frame of the job shop scheduling. Big scale problems demand heuristic algorithms for searching feasible solutions under an optimization criterion. In the case under examination all the constraints and criterion for optimization can be represented by linear equations. At the same time the number of tasks and machines and the time frame are not too large. In this framework the time frame $[0, L]$ is discretized into $t = 1, \ldots, T$ disjoint sub-intervals which have the same length Δt, thus the sub-interval $t = 1$ covers the time frame $[0, \Delta t]$, the sub-interval $t = 2$ covers the time frame $[\Delta, 2\Delta t]$ and so on as far as $t = T$.

Under this considerations, the problem can be dealt by a linear integer formulation based on binary variables related to decision-making process. This formulation is suitable for using a modular integer programming techniques [30].

Nevertheless, in case a long time scheduling calculation is necessary, the problem can split out into separate time intervals, proceeding independently with the calculation of each optima and finally joining chronologically the collection of sub-scheduling. Although the joining together of all the partial sub-scheduling doesn't give the global optimum, it can give a useful feasible solution.

3.1 Variables Model

First we define the integer binary variables used to make the decisions about choosing a machine for processing a task in a particular time interval. Particularly we choose the variables $X_{ijkt} \in \{0, 1\}$, where $X_{ijkt} = 1$ if O_{ij} compound starts being processed by machine M_k at time t and $X_{ijkt} = 0$ otherwise. The ranges of the sub-indexes of variables X_{ijkt} are $i = 1, \ldots, I$, $j = 1, \ldots, l_i$, $k = 1, \ldots, K$ and $t = 1, \ldots, T$.

3.2 Constraints

A set of linear constraints have been designed in order to guarantee the fulfillment of the conditions imposed by the particular industrial environment considered here.

(1) *Each compound of final stage type has to be processed once and only once by any one of the machines available to do it.*

This set of constraint together with the conditions imposed by the flow of supply chain, Eq. (2), will guarantee the manufacturing of all the necessary products for fulfilling the demand. Thereby, the products out of stock, which belong to any demanded task chain has to be processed. For each one of these final tasks, O_{il_i}, $i = 1, \ldots, I$, all the possible starting point have to be explored along the time frame, $1 \leq t \leq T - d_{ijk} + 1$.

$$\sum_{k \in \Lambda_{il_i}} \sum_{t=S_k}^{r_{ik}} X_{il_ikt} = 1, \quad \forall i = 1, \ldots, I. \tag{1}$$

where $\{r_{ik} = \min p_i, E_k - d_{ijk} + 1\}$ is the last moment for starting the compound O_{il_i} and fulfilling the deadline of the sequence as well as timetable availability of the machine k.

The number of this type of constraints is not more than I.

(2) *Each machine only can process one task at the same time.*

This set of constraint is repeated for each task $O_{ij} \in \Phi_{\mathbf{k}}$ compatible with the machine k while this is in service and available to deal with the task, $S_k \leq t \leq E_k - d_{ijk} + 1$. If the task O_{ij} starts being processed by the machine k at time t, then none of the other task compatible, $O_{i'j'} \in \mathbf{\Phi_k}$, could be processed by machine M_k until O_{ij} ends at $t' = t + d_{ijk}$. In this set of constraint only tasks compatible with machine M_k are involved.

$$\sum_{\substack{O_{i'j'} \in \mathbf{\Phi_k}, \\ O_{i'j'} \neq O_{ij}}} \sum_{t'=t}^{t+d_{ijk}+h_{iji'j'}-1} X_{i'j'kt'} + \sum_{t'=t+1}^{t+d_{ijk}-1} X_{ijkt'} \leq L(1 - X_{ijkt}),$$

$$\forall k = 1, \ldots, K,\ \forall O_{ij} \in \mathbf{\Phi_k},\ \forall S_k \leq t \leq E_k - d_{ijk} + 1. \quad (2)$$

where L is a constant large enough, $L > \max_k |\mathbf{\Phi_k}| \cdot \max_{i,j,k} |d_{ijk}|$. Here $|\cdot|$ represents the cardinal of a set. This set of constraint is repeated for each time step while the machine M_k is in service, $S_k \leq t \leq E_k$, therefore the number of constraints of this type is around $(E_k - S_k) \times \sum_k |\mathbf{\Phi_k}|$.

(3) *The jobs have to be finished within a set time or deadline.*

Once a deadline p_i has defined for some jobs, each of them will be finished not later than this time,

$$\sum_{k \in \mathbf{\Lambda_{il_i}}} \sum_{t=S_k}^{E_k} t X_{il_ikt} \leq p_i, \quad \forall i = 1, \ldots, I. \quad (3)$$

This constraint may induce the infeasibility of the problem in case of wrong estimation of the number of jobs the deadlines and the power of the machines. However, sometimes these constraints are substitute by a penalty on the objective function according to the tardiness or delay in jobs finishing.

The number of this type of constraints is not more than I.

(4) *The management of the supply chain has to be guaranteed.*

Because of the discretization in time and the binary variables suggested, the formulation of this job shop problem can be significantly simplified. The most complex constraint corresponds to the supply flow. Before accomplishing a task the existence in stock of certain quantity of the previous compound has to be guaranteed. At this point the compound to be processed O_{ij}, the previous one O_{ij-1} and occasionally the competitors will be involved. The competitors of the compound O_{ij} are other compounds of second stage which also use O_{ij-1} as raw material. The set of competitors associated to compound O_{ij} will be

identified as $\mathbf{C_{ij}}$.

$$\sum_{k\in\mathbf{\Lambda_{ij}}} X_{ijkt} \le S_{ij-1} + \sum_{k'\in\mathbf{\Lambda_{ij-1}}} \sum_{t'=1}^{t-d_{ij-1k}} X_{ij-1k'} - \sum_{O_{i'j'}\in\mathbf{C_{ij}}} \sum_{k''\in\mathbf{\Lambda_{i'j'}}} \sum_{t''=1}^{t} X_{i'j'k''},$$

$$\forall i = 1,\ldots,I,\ j = 1,\ldots,l_i,\ \forall t = 1,\ldots,T - \min_{k\in\mathbf{\Lambda_{ij}}} d_{ijk} + 1. \tag{4}$$

In summary, the sum of the raw material stored at the beginning of the simulation S_{ij-1} plus the raw material produced along the simulation, but earlier than the starting of O_{ij} minus the raw material consumed by the competitors $O_{i'j'} \in \mathbf{C_{ij}}$ will not be less than the quantity of O_{ij} which will start being processed.

The number of this type of constraints is around $I \times ml \times T$, where ml is the average of the number of compounds included in one job.

3.3 Objective Function

The simulations implemented here have been oriented to schedule an efficient sequential assignment of jobs to machines. The efficiency of the performance can be defined and measure in different ways. In our first option we have given priority to minimize the global idle time of the set of machines. Thereby, to find the minimum of the sum of idle time or gaps in the machines is equivalent to find the minimum of the sum of starting time of each of the jobs. It is straightforward to set out a linear objective function representing this measurement,

$$f_1(X) = \sum_{i=1}^{I} \sum_{j=1}^{l_i} \sum_{k\in\mathbf{\Lambda_{ij}}} \sum_{t=1}^{r_{ik}} t X_{ijkt}. \tag{5}$$

where $r_{ik} = \min\{p_i, E_k - d_{ijk} + 1\}$ as in Eq. (1). However, in case of choosing the global ending time of the group of jobs as minimization criterion, then, an integer variable E has to be defined by an additional group of constraints. E cannot be less than each of the ending times of the jobs assigned to any machine,

$$E \ge \sum_{k\in\mathbf{\Lambda_{il_i}}} \sum_{t=S_k}^{E_k - d_{il_ik}+1} (t + d_{il_ik} - 1) X_{il_ikt}, \quad i = 1,\ldots,I. \tag{6}$$

In this case, the objective function to be minimized can be represented by the following brief expression,

$$f_2(X) = E. \tag{7}$$

In the literature a wide variety of suggestions for the objective function can be found, depending on the magnitudes to be under control. For example an objective function could be addressed to the tardiness accumulated by the jobs. The tardiness N_i of the job i is defined as the positive part of the time exceeding the deadline, $N_i = |FJ_{il_i} - p_i|^+$. It means that N_i is greater than 0 only when $FJ_{il_i} - p_i > 0$ and 0 otherwise $N_i = 0$. This quantity is nonlinear but anyway it can be introduce in the linear scheme by a simple set of constraints. Other works as [31] have explored multiobjective formulations for the job shop scheduling problem. But in practice the balance between the different contributions on the objective function is very subjective and increases significantly the computation cost.

4 A Sample Case for Scheduling

Several simulations for making a schedule with the assignments of machines to jobs have been implemented. The objective is to verify and measure the efficiency of the mathematical model. The approach has been developed using C++ programming language and two different optimization tools or solvers. On the one hand, the very well known CPLEX [17] commercial software has been used. On the other hand, COIN libraries [23], one of the most powerful open codes for optimization, has been used. The implementation of these computing tools has the advantage of grouping the variables in one dimensional array Z, vanishing the unused variables. The definition of the variables is correlative from X_{111S_1} to $X_{Il_lKE1_K}$, but only for the compatible compounds O_{ij} and Machines M_k. It means that we do not consider all the variables associated to the pairs (O_{ij}, M_k) with a 0 value in a matrix similar to the one shown in Table 1 in the particular problem to be solved.

One could think on considering all the possible variables, including the corresponding to incompatible pairs (O_{ij}, M_k), and the complete time frame $[1, T]$ in order to write the constraints (1)–(4) in an straight forward manner. The code would have to incorporate a new constraint which would force the sum of all the unused variables to be zero. The disadvantage of this formulation lies with the unnecessary growth of the dimension of the problem. It may result in the eventually collapse of the solver due to the size of the problem and the dimensions of the arrays involved. In order to fit the dimension of the model to the real indispensable set of variables and constraints, it is preferable to use only the variables associate to compatible states. The strategy of setting a one dimension array of correlative list of variables is computationally efficient but makes difficult the reading of the constraints. A strategy to make the formulation more comprehensive involve creating a translation function that links each index set (i, j, k, t) corresponding to (O_{ij}, M_k, t) to a

particular index of the array Z. This procedure makes much more comprehensive the implementation of the constraints in a recursive way taking advantage of the symmetry of the problem. However, modern interfaces implemented in languages as Python incorporates the possibility of working with sets of elements, which makes the code very similar to the classical mathematical formulation of the constraints.

In this work several simulations managing different number of jobs, from 4 to 6, have been performed. Each job is made of 3–5 compounds. The performance obtained by the commercial CPLEX solver is always leading the one obtained by COIN solver. The latter one has the limit of 80×10^6 on the number of elements of the matrix, indeed variables multiplied by constraints. It means that orders larger that six jobs considered to be scheduled in 15 h will not be feasible for COIN, while CPLEX can tackle larger problems.

The schedule obtained by COIN solver for each machine on a group of 7, which are installed in a tire factory, is shown in Table 2. Six jobs have been considered, each one made of 5, 4, 5, 3, 4 and 5 rubber compounds respectively. In total, there are 26 tasks to be assigned to 7 multipurpose machines, M_1, M_2,. . .,M_7. The average number of machines compatible with each compound are 2.5 and the reciprocal average number of compounds compatible to be processed by each machine is 9.3. The scheduling obtained by CPLEX commercially licensed solver under the same environment is shown in Table 3. A time limit of 300 s has been applied to the computation in both cases. We are well aware that the scheduling obtained doesn't reach the absolute optimum, but the branch and bound algorithm here takes

Table 2 A scheduling for 7 machines and 6 product chains over 90 time intervals performed by COIN solver after 2 min of running branch and bound algorithm. Each column indicates the sequence of compounds and starting times assigned to a machine, including gaps or idle time and technical stops

	M_1	M_2	M_3	M_4	M_5	M_6	M_7
	gap = 10	gap = 42	gap = 14	gap = 30	gap = 18	gap = 5	O_{31} $t = 1$
	O_{33} $t = 11$	stop = 12	O_{34} $t = 15$	O_{22} $t = 31$	O_{35} $t = 19$	O_{32} $t = 6$	O_{61} $t = 14$
	O_{62} $t = 20$	O_{43} $t = 55$	gap = 5	gap = 32	gap = 12	gap = 8	O_{21} $t = 26$
	O_{41} $t = 31$	gap = 4	O_{64} $t = 29$	O_{53} $t = 76$	O_{24} $t = 45$	O_{63} $t = 26$	O_{51} $t = 39$
	gap = 18	O_{15} $t = 69$	O_{23} $t = 40$		O_{65} $t = 58$	O_{42} $t = 38$	O_{11} $t = 52$
	O_{13} $t = 60$	gap = 1	gap = 15			gap = 8	
	O_{52} $t = 71$	O_{54} $t = 81$	O_{14} $t = 65$			O_{12} $t = 56$	
End time	$t_1 = 83$	$t_2 = 92$	$t_3 = 75$	$t_4 = 88$	$t_5 = 71$	$t_6 = 67$	$t_7 = 63$
Sum of gaps	$g_1 = 28$	$g_2 = 47$	$g_3 = 34$	$g_4 = 62$	$g_5 = 30$	$g_6 = 21$	$g_7 = 0$

Table 3 A scheduling for 7 machines and 6 product chains over 90 time intervals performed by CPLEX solver after 2 min of running branch and bound algorithm. Each column indicates the sequence of compounds and starting times assigned to a machine, including gaps or idle time and technical stops

	M_1	M_2	M_3	M_4	M_5	M_6	M_7
	O_{41} $t = 0$	gap = 8	gap = 4	gap = 63	gap = 19	gap = 5	O_{61} $t = 1$
	O_{63} $t = 11$	O_{43} $t = 9$	O_{42} $t = 5$	O_{34} $t = 64$	O_{65} $t = 20$	O_{62} $t = 6$	O_{11} $t = 14$
	O_{13} $t = 22$	gap = 11	gap = 1	gap = 28	gap = 6	gap = 2	O_{21} $t = 26$
	gap = 2	O_{15} $t = 32$	O_{64} $t = 16$		O_{24} $t = 40$	O_{12} $t = 19$	O_{51} $t = 39$
	O_{23} $t = 35$	stop = 12	gap = 2		O_{54} $t = 54$	gap = 2	O_{31} $t = 52$
	gap = 10	gap = 16	O_{14} $t = 27$			O_{22} $t = 31$	
	O_{32} $t = 56$	O_{35} $t = 71$	gap = 11			gap = 1	
			O_{53} $t = 48$			O_{52} $t = 44$	
			gap = 2				
			O_{33} $t = 60$				
End time	$t_1 = 66$	$t_2 = 80$	$t_3 = 69$	$t_4 = 76$	$t_5 = 67$	$t_6 = 54$	$t_7 = 63$
Accumulated gaps	$g_1 = 12$	$g_2 = 35$	$g_3 = 18$	$g_4 = 63$	$g_5 = 25$	$g_6 = 10$	$g_7 = 0$

unacceptable long time to converge. In such conditions an acceptable as well as fast solution is better than a slow solution, even though it is close from the global optimum. Only in simulations developed for two jobs by COIN and for four jobs by CPLEX the global optimum was found before 500 s of computing time.

Within the industrial context analyzed in this study case, a reasonable measure to compare the output given by COIN and by CPLEX is the gaps or idle time accumulated by the seven machines and the latest of the end times of the schedules. In one hand, the idle time accumulated by seven machines in the scheduling calculated by COIN solver is $\sum_{i=1}^{7} g_i = 222$ and the last of the end times $t_2 = 92$. On the other hand, the idle time accumulated by seven machines in the scheduling calculated by CPLEX solver is $\sum_{i=1}^{7} g_i = 163$ and the last of the end times $t_2 = 80$. The greater performance of the results given by CPLEX solver can be very clearly observed.

In many simulations we have observed that CPLEX solver takes advantage of the possibility of overlapping consecutive compounds of the same job very efficiently when early feasible solutions are calculated. This phenomenon can be observed in

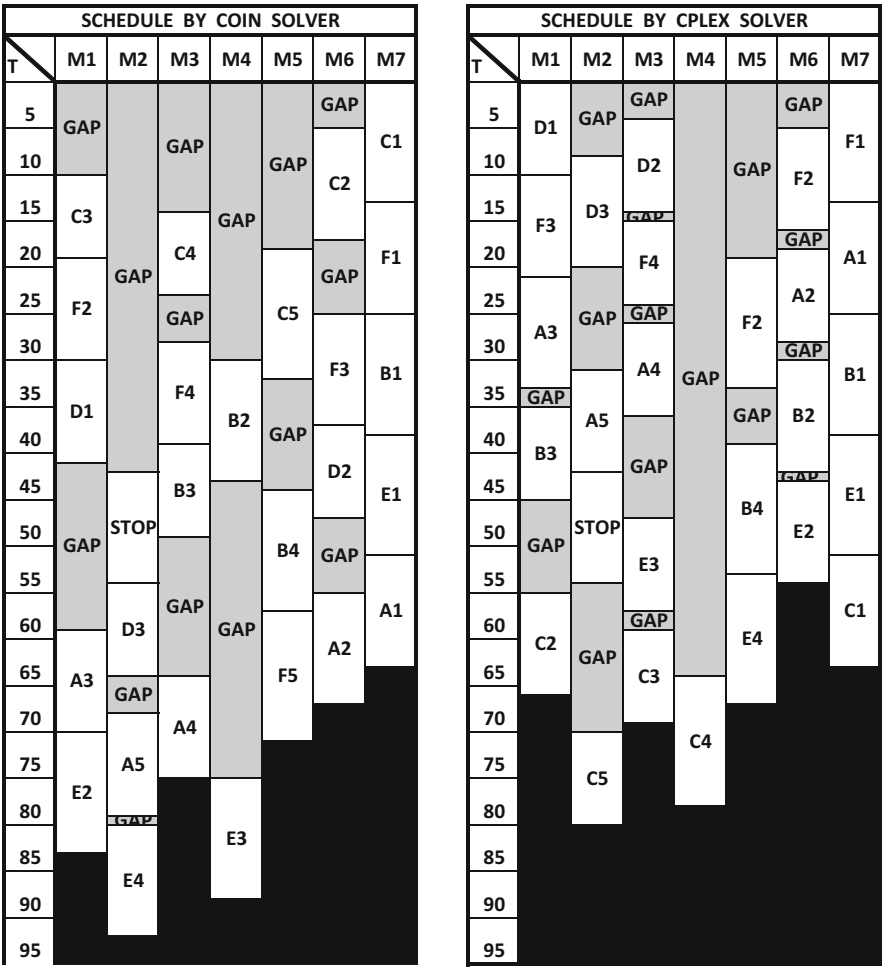

Fig. 2 A comparison of a scheduling of six sequences of jobs, A, B, C, D, E and F, assigned to seven machines, M1,. . ., M7, and computed independently by COIN solver and by CPLEX solver. The jth job of each sequence is labeled by "j" digit. The range of the discretized time frame is $t = 1, 2, \ldots, 95$. Grey cells correspond to idle time periods of the machines

the staggered manner of the manufacturing of some jobs in the graphical scheduling shown in Fig. 2.

Comparison between CPLEX solver performance and COIN solver performance have been made for sets of 4, 6 and 8 jobs and different time limits applied to computation, indeed, from 50 to 600 s. Repeatedly the performance shown by CPLEX commercially licensed solver is significantly better than the one given by COIN free software. Anyway the difference in the performance between two solvers is bigger as shorter is the time limit of the simulation. However, when the time

limit applied to the simulation increases, then the performance of the scheduling calculated by two solvers approach. It means that the algorithms to find a the initial feasible solution and the strategies to apply branch and bound cutting plane is more efficient in the case of CPLEX, but in so far as simulation is prolonged, solutions given by COIN improve significantly and also can be considered as a good option for practical scheduling.

5 Conclusions

A tire manufacturer manages a long catalog of rubber layers because each type of vehicle needs tires conforming to particular specifications. A rubber compound ready for assembling is made of an ordered sequence of compounds. The manufacturing of those compounds is a very suitable process to be optimized by operations research techniques, because of the flow conditions and the multipurpose character of the machines. Companies dedicated to producing tires very often need to update the scheduling that determinate the tasks to be accomplished by the machines. The incidents arising in the machines and the management of new urgent incoming orders are frequently the reason for this. In the case of recalculation of the scheduling, the promptness for giving an optimal solution for a small set of sequences of compounds, for example 4, is more useful than a large scheduling that needs a long computing time. In this scenario the use of discrete integer linear programming formulation for modeling the requirements of the scheduling in terms of constraints and objective function is simple and efficient.

Several types of simulations characteristic of each one of the case studies have been performed in this work. In our experiments, commercial solvers running on a personal computer can deal with groups of 4 jobs integrated by around 25 compounds, while free solvers can deal with groups of 3 jobs integrated by around 15 compounds. In both cases the computing time doesn't exceeds 4 min and the outputs give rise to the scheduling fitting the global minimum of the idle time of the machines. Long-scale production scheduling can be elaborated by joining consecutive partial scheduling made for subgroups of jobs. A strategy consisting of dividing the problem and joining partial schedules doesn't guarantee the global optimum, but anyway it lead to practical and fast feasible solution. The partner company has recognized that the response time for recalculating the schedule when the conditions suddenly change, has been reduced around four times. The results of this work show the successfully contribution of the operations research tools in the improvement of the profits quality of existing industrial processes.

As future research, other real elements concerning the manufacturing dynamics of the company can be considered. For example the case of machines blocked for performing pre-programmed tasks or the case of programmed maintenance stop of some machine into the time frame of the simulation. Those new considerations as well as the possibility of planning simulations for larger groups of around 10 jobs makes the problem more complex. In that scenario integer programming

formulation is insufficient for giving rise to successful solutions and mixed integer programming could be more suitable for dealing with the problem.

Acknowledgments Mikel Lezaun and Carlos Gorria has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 777778 (MATHROCKS), the Project of the Spanish Ministry of Economy and Competitiveness with reference MTM2016-76329-R (AEI/FEDER, EU) and the Basque Government through the two Elkartek projects ArgIA (KK-2019-00068) and MATHEO (KK-2019-00085), and the Consolidated Research Group MATHMODE (IT1294-19), given by the Department of Education.

References

1. Akers, S.B.: A graphical approach to production scheduling problems. Oper. Res. **4**(2), 244–245 (1956). https://doi.org/10.1287/opre.4.2.244
2. Applegate, D., Cook, W.: A computational study of the job-shop scheduling problem. Informs J. Comput. **3**(2), 149–156 (1991). https://doi.org/10.1287/ijoc.3.2.149
3. Bazaraa, M.S., Jarvis, J.J., Sherali, H.D.: Linear Programming and Network Flows. Wiley, Hoboken (2010). https://doi.org/10.1016/0377-2217(91)90043-U
4. Bierwirth, C., Mattfeld D.C.: Production scheduling and rescheduling with genetic algorithms. Evol. Comput. **7**(1), 1–17 (1999). https://doi.org/10.1162/evco.1999.7.1.1
5. Blazewicz, J., Domschke, W., Pesch, E.: The job shop scheduling problem: conventional and new solution techniques. Eur. J. Oper. Res. **93**(1), 1–33 (1996). https://doi.org/10.1016/0377-2217(95)00362-2
6. Brucker, P., Schlie, R.: Job-shop scheduling with multi-purpose machines. Computing **45**(4), 369–375 (1990). https://doi.org/10.1007/BF02238804
7. Brucker, P., Jurisch, B., Sievers, B.: A branch and bound algorithm for the job-shop scheduling problem. Discrete. Appl. Math. **49**(1), 107–127 (1994). https://doi.org/10.1016/0166-218X(94)90204-6
8. Carlier, J., Pinson, E.: An algorithm for solving the job shop problem. Manage. Sci. **35**(29), 164–176 (1989). https://doi.org/10.1287/mnsc.35.2.164
9. Carlier, J., Pinson, E.: A practical use of Jackson's preemptive schedule for solving the job-shop problem. Ann. Oper. Res. **26**, 269–287 (1990). https://doi.org/10.1007/BF03543071
10. Darayi, M., Eskandari, H., Geiger, C.D.: Using simulation-based optimization to improve performance at a tire manufacturing company. QScience Connect **2013**(1), 1–12 (2013). https://doi.org/10.5339/connect.2013.13
11. Dauzère-Pérès, S., Paulli, J.: An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Ann. Oper. Res. **70**, 281–306 (1997). https://doi.org/10.1023/A:1018930406487
12. Degraeve, Z., Schrage, L.: A tire production scheduling system for Bridgestone/Firestone off-the-road. Oper. Res. **45**(6), 789–796 (1997). https://doi.org/10.1287/opre.45.6.789
13. Framinan, J.M., Leisten, R., Ruiz, R.: Manufacturing Scheduling Systems. Springer, London (2014). https://doi.org/10.1007/978-1-4471-6272-8
14. French, S.: Sequencing and scheduling. An introduction to the mathematics of the job-shop. Wiley, New York (1982). https://doi.org/10.1002/net.3230130218
15. Giffler, B., Thompson, G.L.: Algorithms for solving production scheduling problems. Oper. Res. **8**(4), 487–503 (1960). https://doi.org/10.1287/opre.8.4.487
16. Graves, S.C.: A review of production scheduling. Oper. Res. **29**(4), 646–675 (1981). https://doi.org/10.1287/opre.29.4.646

17. High-performance mathematical programming commercial solver for linear programming, mixed-integer programming and quadratic programming. https://www.ibm.com/analytics/cplex-optimizer
18. Hurink, J, Jurish, B, Thole, M.: Tabu search for the job shop scheduling problem with multi-purpose machines. OR-Spektrum **15**(4), 205–215 (1994). https://doi.org/10.1007/BF01719451
19. Jackson J.R.: Simulation research on job shop production. Nav. Res. Logist. Q. **4**(4), 287–295 (1957). https://doi.org/10.1002/nav.3800040404
20. Jurisch, B.: Lower bounds for the job-shop scheduling problem on multi-purpose machines. Discrete. Appl. Math. **58**(2), 145–156 (1995). https://doi.org/10.1016/0166-218X(93)E0124-H
21. Kim, H.H., Kim, D.G., Choi, J.Y., Park, S.C.: Tire mixing process scheduling using particle swarm optimization. Comput. Ind. Eng. **110**, 333–343 (2017). https://doi.org/10.1016/j.cie.2017.06.012
22. Lageweg, B.J., Lenstra, J.K., Rinnooy Kan, A.H.G.: Job shop scheduling by implicit enumeration. Manag. Sci. **24**(4), 441–450 (1977). https://doi.org/10.1287/mnsc.24.4.441
23. Open source interface and libraries for the operations research community. https://www.coin-or.org/
24. Ovacik, I. M., Uzsoy, R.: Decomposition methods for complex factory scheduling problems. Springer Science & Business Media, New York (1997). https://doi.org/10.1007/978-1-4615-6329-7
25. Özgüven, C., Yavuz, Y., Özbakir, L.: Mixed integer goal programming models for the flexible job-shop scheduling problems with separable and non-separable sequence dependent setup times. Appl. Math. Model. **36**, 846–858 (2012). https://doi.org/10.1016/j.apm.2011.07.037
26. Peng, C., Wu, G., Liao, T.W., Wang, H.: Research on multi-agent genetic algorithm based on tabu search for the job shop scheduling problem. PLoS One **14**(9), e0223182 (2019). https://doi.org/10.1371/journal.pone.0223182
27. Pezzella, F., Morganti, G.: A genetic algorithm for the flexible job-shop scheduling problem. Comput. Oper. Res. **35**(10), 3202-3212 (2008). https://doi.org/10.1016/j.cor.2007.02.014
28. Pinedo, M.L.: Scheduling. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-319-26580-3
29. Tamssaoueta, K., Dauzère-Pérès, S., Yugmaa, C.: Metaheuristics for the job-shop scheduling problem with machine availability constraints. Comput. Ind. Eng. **125**, 1–8 (2018). https://doi.org/10.1016/j.cie.2018.08.008
30. Wolsey, L.A., Nemhauser, G.L.: Integer and combinatorial optimization. Wiley-Interscience, New York (1999). https://doi.org/10.1057/jors.1990.26
31. Xia, W., Wu, Z.: An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Comput. Ind. Eng. **48**(2), 409–425 (2005). https://doi.org/10.1016/j.cie.2005.01.018