The International Journal of Advanced Manufacturing Technology (2021) 115:915–925
https://doi.org/10.1007/s00170-020-06176-y

ORIGINAL ARTICLE

Workforce influence on manufacturing machines schedules

Pierpaolo Caricato[1] · Antonio Grieco[1] · Anna Arigliano[1] · Luciano Rondone[2]

Received: 31 August 2020 / Accepted: 24 September 2020 / Published online: 12 October 2020

Abstract
This study addresses a parallel machines scheduling problem with sequence-dependent setup times and additional resource constraints related to workforce management. In most industrial cases, the execution of jobs requires the involvement of human resources in addition to machines: this work addresses the many complications due to workforce-specific issues that arise in a real industrial application. This is achieved separating the complex yet classical parallel machines scheduling problem with sequence-dependent setup times from the additional human resources planning problem: the former is formulated and solved through constraint programming, while an ad hoc procedure is provided for the latter. An Italian specialized firm, Prosino S.r.l., provides the industrial case to both validate the adequacy of the adopted method to the actual industrial problem and test the effectiveness of the proposed approach. Computational results obtained over six months of experimentation at the partner firm are presented.

Keywords Production scheduling · Resource-constrained parallel machine scheduling problem · Sequence-dependent setup times · Additional limited resources · Workforce

1 Introduction

While the theoretical management of a production system can focus on specific aspects, congenial to a better isolation of mathematical properties of the production planning problem, a realistic management of a production system cannot avoid simultaneously dealing with multiple issues, such as multiple stages and machines, human resources, sequence-dependent setup times, and workers skills. With this in mind, the paper addresses a real industrial production planning problem, coming from *Prosino S.r.l.*, an Italian firm that manufactures spinning and twisting rings used in high precision bearings. A hierarchical approach, based on a hybrid constraint programming model, to solve the short and midterm production planning problem the firm addresses is presented.

The case study can be widely framed in the flexible flow shop manufacturing set of problems: the classical flexible flow shop problem (FFSP) considers N stages in series and each stage k includes m_k identical parallel machines; in other words, a FFSP can be seen as the combination of a flow shop problem with a parallel machine scheduling problem (see [1]).

The literature provides many contributions on FFSPs, as many real applications can be effectively formalized using this model. These problems are extremely hard to solve, as thoroughly discussed in [2] and many contributions can be found both about the original problem and on its variants, but only few works consider limited resources, mainly human resources, in addition to the available machines. Among these ones, [3] address a variant of the FSP, i.e., multiple stages but with a single machine per stage, where the objective is to assign operators to machines when the number of workers is less than the number of machines: the authors show that assigning operators and simultaneously building a schedule is an NP-hard problem. A contribution

✉ Pierpaolo Caricato
pierpaolo.caricato@unisalento.it

Antonio Grieco
antonio.grieco@unisalento.it

Anna Arigliano
anna.arigliano@unisalento.it

Luciano Rondone
lrondone@prosino.com

1 Dipartimento di Ingegneria dell'Innovazione, Universitità del Salento, Lecce, Italy

2 Prosino S.r.l., Grignasco, Italy

that more closely matches the test case is given by [4]: here, a FFSP with worker flexibility is considered, where in each stage the number of workers equals the number of machines. Hence, to the authors' knowledge, there is no contribution that considers a FFSP with additional workers as limited resources, and specifically with less workers than the number of machines available at each stage.

The proposed approach adopts a hierarchical procedure in which, at each iteration, a constraint programming model is solved to manage the classical job sequencing part of the problem at hand, while a constructive heuristic addresses the peculiar aspects related with workforce management issues. Updates on release dates and precedence constraints between consecutive iterations allow the coordination of the overall production plan among the different stages. This work is hence focused on the single-stage resolution, i.e., the scheduling problem on parallel machines with additional resources and sequence-dependent setup times, while also presenting the overall hierarchical procedure. The paper is organized as follows: in Section 2, a review of the related works is discussed; Section 3 describes the industrial problem; while Section 4 presents the hybrid constraint programming procedure used to solve the single-stage problem. An experimental study is then presented in Section 5. Finally, conclusions are pointed out in Section 6.

2 Related works

Production scheduling problems involving multiple stages and parallel machines have been covered by numerous contributions over the last years. In most real cases, the execution of operations in manufacturing firms requires additional resources in addition to machines. In these situations, both the job scheduling and the allocation of additional resources to machines should be taken into account. According to [5] and [6], the additional resources can be classified as:

- Renewable (non-storable): resources that can be used throughout the project and characterized by a capacity that is limited all the time (examples of such type include manpower and tools);
- Non-renewable (storable): resources that can be consumed by jobs, eventually leading to their unavailability at a certain point (e.g., raw materials, fuels);
- Doubly constrained: resources that are constrained both in usage and in total quantity.

Furthermore, it is possible to distinguish between two types of resource allocation [7, 8]:

- Static: when the allocation of additional resources to machines is not limited, but it has to remain the same throughout the scheduling horizon;
- Dynamic: a more general case, when a resource can be allocated on different machines at different times of the schedule.

A detailed review on *parallel machines with additional resources problems* is provided in [6]. The survey follows a classification scheme based on five main categories: machine environment, additional resources, objective functions, complexity results, and solution methods. According to the problem classification provided by the authors, the problem treated in this paper can be classified as a version of the *resource-constrained parallel machine scheduling problem* (RCPMSP), but to the best of the authors' knowledge, no work that addresses both resource allocations and setups is available in the literature; however, the following papers address scheduling problems with additional renewable resources, with more attention to the management of human resources.

In [9], heuristic solutions are proposed to solve the scheduling problem of a set of jobs on identical parallel machines where the workers are a critical resource. Each operator is associated with several working modes. The decision deals with the choice of a working mode for each operator in a context where different working modes induce different productivity rates on each machine.

Chen [10] studies the parallel machine scheduling problem involving job processing and resource allocation, where the job processing time is a non-increasing function with the allocated resources. The paper introduces a column generation–based branch-and-bound method in order to solve two types of problems: the minimization of the total completion time and the minimization of the weighted number of tardy jobs.

Hu [11] minimizes the total flow time in a parallel machine system assigning jobs and workers to the machines. The author proposes a heuristic solution with two sequential procedures: the former assigning the jobs and the latter the workers to the machines.

In [12], a flexible model to address integrated employee timetabling and production scheduling is proposed. The authors exploit the flexibility of constraint programming modeling to represent complex relationships between schedules and activity requests. A hybrid CP-LP exact method is presented in order to solve a lexicographic makespan and employee cost optimization problem.

In many cases, the resources are all of a single type: [13], instead, analyze the complexity of different versions of the general $PD_m|res\lambda\sigma\rho|C_{max}$ problem, i.e., a scheduling problem for m parallel machines under resource constraints, where λ resource types are to be considered, the size of each resource does not exceed σ, and each job consumes no more than ρ units of resources.

The complexity of the problem has also led many authors to break down the problem into simpler subproblems and others to consider heuristic algorithms. In fact, recent works deal with scheduling problem of unrelated parallel machines, using the following pattern:

- A mixed integer programming (MIP) exact formulation to address small-size instances of the problem;
- Heuristics/metaheuristics approaches to solve lager instances.

Following this pattern, [14] use a MIP formulation for an unrelated parallel machine scheduling problem with limited resources. An operator is needed for each setup activity between two consecutive jobs on the same machine. However, this model is able to solve to optimality only small-size instances. In order to cope with larger instances, a genetic algorithm is presented.

Afzalirad and Rezaeian [15] address an unrelated parallel machines scheduling problem with resource constraints, sequence-dependent setup times, different release dates, machine eligibility, and precedence constraints. Their work models a real case study, i.e., the block erection scheduling problem in a shipyard. A first MIP formulation is proposed that is able to solve only small instances of the strongly NP-hard problem addressed, along with two new metaheuristic algorithms to provide good solutions for larger instances. Afzalirad and Shafipour [16] treat a simplified version of the problem, where precedence and release date constraints are not considered and setup times are assumed to be part of the processing times.

[17] is a new study based on the same type of problem defined by [15], where a feasible schedule must also consider constraints on release dates, limited additional resources, and sequence-dependent setup times. v types of resources are considered and, for each type, a limited number of units is available. The authors propose two approaches: an exact mixed-integer linear programming model (MILP) and a two-stage hybrid metaheuristics based on variable neighborhood search and simulated annealing.

Resource-constrained project scheduling (RCPS) problems can be seen as akin to RCPMSPs, since they also model scheduling problems subject to resource constraints. In fact, project scheduling can be considered, under specific conditions, as a parallel machine scheduling problem with precedence constraints among activities. In [18], an extensive overview of approaches and solution categories for the RCPSP is provided: the author explores exact, heuristic, and metaheuristic algorithms for such problems. A review of the books [19, 20] and [21] is provided in [22], covering a wide range of problems related to project scheduling. In particular, [20] addresses hard resource-constrained project scheduling problems using constraint propagation techniques.

3 The industrial problem

The test case comes from a production plant that includes N floors that can be associated with the stages of a FFSP. Unlike the classical FFSP, however, additional resources are required in some stages, namely human resources to supervise machines' operations. Constraints associated with workers make the problem more complex than the theoretical cases studied in the literature; therefore, the problem requires a customized solution model that allows makespan minimization according to all required constraints.

Each worker is assigned to a specific department and the number of workers per shop floor is always less than the number of parallel machines available in the floor. The time horizon is partitioned into time shifts and the availability of additional resources depends on several aspects:

- Worker skills: each worker has certain capabilities to carry out some machining or tooling tasks and he can only be assigned to the operations for which he is qualified.
- Calendars: the presence of each worker is specified in a personal calendar specific for each worker.
- Parallel working mode: each operator can supervise more than one machine simultaneously depending on some conditions:
    1. The adjacency of the machines supervised by the operator;
    2. The operation type assigned to the work centers candidate to work in parallel;
    3. The number of working shift during which each worker can operate in parallel mode can be limited by company agreements with the workers.
- Teams: groups of several workers can be formed to allow parallel supervision of more than two machines, also overcoming some of the previous limitations (e.g., three operators per four machines).

The overall production planning problem is decomposed through an iterative procedure consisting of N steps, as many as the company floors (i.e., the number of stages in the flexible flow shop). Each run k processes all operations belong to the same stage k: the solution to the scheduling and workforce assignment problem of the current stage provides release dates and precedence constraints for the following stage. The focus of this work is the solution of the planning problem solve in each stage.

4 Single-stage planning

The single-stage planning problem requires the scheduling of a set of n independent jobs $J = \{j_1, j_2, \dots, j_n\}$ on a set of m parallel machines $M = \{m_1, m_2, \dots, m_m\}$ with sequence-dependent setup times, s_{ijm} $\forall i, j \in J$ and $m \in M$. A summary of the notation used is reported in Table 1.

A job j can be either available for processing at time 0 or have a release date ρ_j deriving from the solution of the planning problem of the previous stage. Each job can be processed on a subset of compatible machines, $M_j = \{m_1, \dots, m_{n_j}\}$. Each machine can process at most one job at a time, and each job cannot be split among several machines. Interrupting the processing of a job is allowed only if a lack of production capacity occurs: i.e., if either the required machine or the supervising worker is not available. Let $W = \{w_1, w_2, \dots, w_l\}$ be the set of workers in the considered stage, where $l < m$: each worker w can/cannot be skilled to work a given job j and can/cannot tooling a given machine m. The number of available workers varies with the shift of the day.

Table 1 Notation

Name	Description
J	The set of all jobs j to be planned
ρ_j	The release date for job j as determined by the solution of the previous single-stage problem ($\rho_j = 0$ if no release date is defined for job j)
M	The set of available machines m
J_m	The set of all jobs j that can be processed by machine $m \in M$
$\{0, 1, \dots, H\}$	Planning horizon
$\Theta = \{\tau_1, \tau_2, \dots, \tau_k\}$	Set of time slots (shifts) that cover the planning horizon, i.e., $\bigcup_{i=1}^{k} \tau_i = \{0, 1, \dots, H\}$
$\tau_i = [\tau_i^s, \tau_i^e]$	Each slot is characterized by a start time τ_i^s and an end time τ_i^e
B_m	Set of break intervals $\{b_{1m}, b_{2m}, \dots, b_{nm}\}$ during which the machine m cannot be used.
S_m	Setup matrix on machine m. The matrix element s_{ijm} is the setup time between the jobs i and j if they are consecutively processed on machine m.
σ_j	Processing speed required by job j
$\bar{\sigma}_{max}$	Speed limit allowed for each pair of machines simultaneously supervised by a single worker

The setup time between two consecutive jobs depends on both the ordered couple of jobs and the machine where the jobs are processed; hence, the sequence-dependent setup time when scheduling the j-th job immediately after the i-th job on machine m is expressed as $s_{ijm} \in S_m$.

Parallel conduction of multiple machines is a key efficiency option that allows a better usage of the available workforce. A single operator can supervise one or two machines (provided that the two machines are adjacent and respect specific technological constraints), while predefined teams of three workers can simultaneously conduct groups of four machines. Hence, we can distinguish three types of "operating modes" for a worker: "single" mode (when the worker supervises only one machine), "parallel" mode (when he simultaneously supervises two machines), and "team" mode (when he works within a team). The adoption of parallel mode for a worker is limited by a day-off rule: if a worker conducts two machines on a day, he must conduct a single machine on the following day. Team mode, on the other hand, can be used everyday and, furthermore, allows the planner to ignore skills, since teams are formed matching workers with assorted capabilities.

Adjacency and technological information needed to define which couples of machines can be conducted by a single worker is coded in terms of predefined groups $G = \{G_1, G_2, \dots, G_{h1}\}$. Groups of four machines that can be supervised by a team of three workers are defined in $\Gamma = \{\Gamma_1, \Gamma_2, \dots, \Gamma_{h2}\}$, where each group consists of 4 machines $\Gamma_i = \{m_{i,1}, m_{i,2}\, m_{i,3}\, m_{i,4}\}$. There is a limited number of teams per stage $T_m = \{T_{m_1}, \dots, T_{m_g}\}$ and each team $T_{m_i} = \{w_{iA}, w_{iB}, w_{iC}\}$ can only be assigned to a group of machines during a work shift.

Each job j is characterized by a working speed, σ_j, given in terms of number of items per hour. The job processing time, p_j, is not affected by the operating mode of the worker, but there is a technological constraint: the sum of the machine speeds simultaneously supervised by the same operator cannot exceed a fixed limit. This limit does not apply for groups of machines conducted by a team.

A worker that starts a job may be replaced by another one in the following shifts, when the processing of a job lasts more than a working shift, with no noteworthy effect nor interruption on the processed job. The objective is to plan the set of jobs to be assigned to each machine, along with the workforce details needed, pursuing the minimization of the makespan.

Since parallel machines scheduling problems with additional resources are known to be NP-hard [6, 23], a heuristic solution is proposed, in particular a hierarchical approach that decomposes the problem into two subproblems to be sequentially solved:

– A job assignment and sequencing problem with sequence-dependent setup times, consisting in the definition of the sequence of jobs on each machine, with the objective to minimize the total setup cost;
– An additional resource allocation problem, where the workers are assigned to machines to conduct the tasks scheduled at the previous step, with the definition of the parallel mode for each worker.

4.1 Constraint programming model

The first subproblem is modeled through constraint programming (CP). Constraint programming is an approach initially developed to model and solve constraint satisfaction problems (CSPs), but it was extended to solve optimization problems as well. A CSP consists in finding values, within finite domains, to be allocated to problem variables, so that all the problem constraints are satisfied [24]. The CP approach consists of two phases: the former is the formalization of the problem in terms of a set of variables with finite domains and a set of constraints that specify which assignments of values to variables are feasible; the latter uses tree search algorithms to systematically explore the possible assignments of values to variables. The search phase combines domain reduction (DRA) and constraint propagation (CPA) techniques to rapidly find a feasible solution or certify that the problem is infeasible.

The formulation presented in this paper adopts the OPL, a formalism to model constraint programming problems presented in [25], which is embedded in the IBM ILOG CPLEX Optimization Studio [26]. Such formal language allows the definition on scheduling specific types of finite domain variables, namely interval variables and sequence variables, along with specific scheduling related constraints that efficiently perform during the search and propagation phase of the solving algorithm. A detailed analysis of the CP approach provided by OPL, with a formal description of the scheduling related variables and constraints, can be found in [27]. Considering the notation given in Table 1, the following model is defined:

$$\min \max_{j \in J}(EndOf(x_j)) \tag{1}$$

$$x_j : IntervalVariable(\rho_j, H) \quad \forall j \in J \tag{2}$$

$$y_{jm} : IntervalVariable(\rho_j, H) \quad \forall m \in M, \quad \forall j \in J_m \tag{3}$$

$$Alternative(x_j, all(y_{jm}{:}\forall m \in M_j)) \quad \forall j \in J \tag{4}$$

$$z_m = Sequence(m, all(y_{jm} : \forall j \in J_m)) \quad \forall m \in M \tag{5}$$

$$NoOverlap(z_m, S_m) \quad \forall m \in M \tag{6}$$

$$Intensity(y_{jm}, B_m) \quad \forall m \in M, \quad \forall j \in J_m \tag{7}$$

$$ForbidStart(y_{jm}, B_m) \quad \forall m \in M, \quad \forall j \in J_m \tag{8}$$

$$ForbidEnd(y_{jm}, B_m) \quad \forall m \in M, \quad \forall j \in J_m \tag{9}$$

The objective (1) is to minimize the makespan, i.e., the end of the latest job that is processed.

An interval variable is defined in Eq. 2 for each job j, with a finite domain $[\rho_j, H]$ for its associated start and end. An interval variable is defined in Eq. 3 for each job j and for each machine m that is compatible with the job, with a finite domain $[\rho_j, H]$ for its associated start and end: this variable represents the possibility to assign the job j to machine m and the consequent start and end times if this decision is taken (i.e., if the variable is "present" in the solution, according to the formalism reported in [26]). Constraint (4) states that each interval variable x_j must be equal to exactly one of the y_{jm} interval variables defined for j, i.e., job j must be processed by a single machine. Equation (5) defines an interval sequence variable z_m for each machine m: an interval sequence decision variable is defined on a set of interval variables and its value represents a total ordering of the interval variables of the set. Any absent (i.e., "not present") interval variable is not considered in the ordering. The set of intervals that are suitable to form a sequence on a given machine m includes all the $y_{jm}, \forall j \in J_m$ interval variables. Constraint (6) does not permit any temporal overlap among tasks assigned to the same machine m. The setup times depend on the work center (machine) and the job type: they are provided by the matrix S_m.

An availability calendar for each machine is given a priori in order to be able to consider the days off and the scheduled maintenances in the scheduling problem. Constraints (7)–(9) are also calendar-related constraints: (7) states that each job j assigned to machine m cannot be processed during any break interval belonging to the machine calendar of m; constraints (8) and (9) avoid that any job j starts or finishes, respectively, during any break interval on machine calendar m.

4.2 Workers allocation

The second subproblem outlined at the beginning of this section is solved using a constructive heuristic that allows to determine a feasible allocation of workers on the machine sequences obtained from the CP subproblem.

The output of the previous step is an input for this procedure: the CP model assigns the jobs to be planned to the available machines and defines the sequence of jobs to be processed on each machine. In addition to the notation provided in Table 1, let $Q = Q_1, \dots Q_m$ be the set of task queues associated with each machine, where $Q_i = (j_{m1}, s_{m1,2}, j_{m2}, s_{m2,3} \dots, j_{mn})$ is the sequence of jobs and setup tasks on machine m as determined by the CP model. It is important to note that each machine sequence includes both the jobs and the setup operations. In this phase, the setups are explicitly considered as jobs because their execution requires the presence of human resources. The macro steps executed by the proposed heuristics are reported in Algorithm 1. It is worth noting that this procedure is generic and valid for both job and setup operations, and the different handling of these two task types is achieved through the skills required by the tasks and the capabilities defined for the workers. The state variables used by the algorithm at each iteration are reported in Table 2. The following subsections describe the behavior of the functions reported in Algorithm 1.

Algorithm 1 Workers allocation algorithm.

```
$i \Leftarrow 1$
while $i \le k$ do
    $\bar{\tau} \Leftarrow T_i$
    $SetTeams()$
    $\{Q_{\sigma(1)}, Q_{\sigma(2)}, \dots, Q_{\sigma(m)}\} \Leftarrow Sort(Q_m)$
    $h \Leftarrow 1$
    while $h \le m$ do
        $Q_{\bar{m}} \Leftarrow Q_{\sigma(h)}$
        if $Q_{\bar{m}} \ne \emptyset$ then $\bar{j} \Leftarrow DeQueue(Q_{\bar{m}})$
            $W_{\bar{j}} \Leftarrow FindWorker(\bar{j})$
            if $W_{\bar{j}} \ne \emptyset$ then $l = 1$, $f \Leftarrow False$
                while $l \le |W_{\bar{j}}| \wedge \neg f$ do
                    $\bar{w} \Leftarrow W_{\bar{j}}[l]$
                    $l \Leftarrow l + 1$
                    if $EvaluateConduction(\bar{w}) > 0$ then
                        $r_{\bar{j}} \Leftarrow SetTask(\bar{j}, \bar{m}, \bar{w}, con)$
                        if $r_{\bar{j}} > 0$ then $\bar{j} \Rightarrow EnQueue(Q_{\bar{m}})$
                        end if
                        $f \Leftarrow True$
                    end if
                end while
            end if
        end if
        $h \Leftarrow h + 1$
    end while
    $i \Leftarrow i + 1$
end while
```

Table 2 State variables

Name	Description
$\bar{j}$	Current job
$\bar{m}$	Current machine
$\bar{\tau}$	Current time shift
$r_{\bar{j}}$	Current residual work of job $\bar{j}$. If the processing of the current job cannot end in the current time shift, $\bar{\tau}$, then it has a residual work greater than 0
$\bar{w}$	Current worker
$W_{\bar{j}}$	Set of possible workers for current job $\bar{j}$

4.2.1 Function Sort()

This function selects the subset of machines available in the current time shift $\bar{\tau}$ (tacking into account the machine calendars) and orders them according to these rules:

1 First available instant time;
2 Largest residual work time on machine.

4.2.2 SetTeams()

This function assigns worker teams to machine groups at the start of the current time shift $\bar{\tau}$. All teams are defined in advance, i.e., its components are always the same throughout the whole planning horizon. The team $T_{m_i} \in T_m$ is assigned to a group $\Gamma_j \in \Gamma$ if the following conditions are verified:

– Each worker $w \in T_{m_i}$ is available at the current time shift $\bar{\tau}$;
– Each $m \in \Gamma_i$ is available at the current time shift and it has a residual work load at least equal to the duration of the time shift;
– The overall residual workload of machine group Γ_i is the largest among the available groups in Γ.

If the above conditions are true, all workers in the T_{m_i} team are assigned to the machine group Γ_i for the entire current time shift and they are excluded from any other assignments during this time period.

4.2.3 Function DeQueue()

Every set Q_i is modeled as a *last in first out* (LIFO) queue. The function DeQueue retrieves the job at the top of the machine stack. At the start of the procedure, Q_i is initialized by pushing the sequence elements in reverse order with respect to the job order on machine i, obtained by the CP solver. If a job is not finished at the end of a time period on a given machine, it will hence be the first to be extracted for the following period when that machine resumes its work.

4.2.4 Function FindWorker()

This function defines a set of workers compatible with the current job, i.e., with the necessary skills to process $\bar{j}$. Moreover, only the workers available in the current time shift, $\bar{\tau}$, are considered. The set is ordered applying the following priority rules and the first worker is returned:

1 Last active worker on the current machine $\bar{m}$. The worker who conducted the machine during the previous job processing, within the same current shift $\bar{\tau}$, has a higher priority.
2 First available instant time of worker.
3 If the current job $\bar{j}$ is not a setup task, the worker without the setup skill has higher priority.
4 Parallel mode. The workers who have already been assigned to another machine are preferred. This rule allows to exploit the parallel conduction form.

4.2.5 EvaluateConduction()

This function determines whether, and in which mode, the current worker $\bar{w}$ can conduct the current job $\bar{j}$ queued on the current machine $\bar{m}$ during the current time shift $\bar{\tau}$. The function returns an integer value chosen among:

0 - No Conduction: the worker capacity has been saturated during the current time shift, hence he cannot start another job. This capacity is time-dependent, since each worker can conduct in parallel on alternate days.
1 - Single Conduction: the worker is available for parallel conduction, but he is not allowed to do it due to the violation of machine constraints. For example, let m_1 be the machine on which the current worker $\bar{w}$ has been assigned to conduct the job j_1 with a working speed σ_{j_1}. Suppose that the function is evaluating the assignment of the job $\bar{j}$ on the current machine $\bar{m}$ with working speed $\sigma_{\bar{j}}$ in the same shift $\bar{\tau}$. If $\sigma_{j_1} + \sigma_{\bar{j}} > \bar{\sigma}_{\max}$ then the worker cannot work in parallel mode. Another cause of impossibility to work in parallel is given by the distance between machines: i.e., if $\nexists G_i \in G$ s.t. $G_i = (m_1, \bar{m})$. In all these cases the function *EvaluateConduction()* returns 1.
2 - Parallel Conduction: any conduction (parallel or single) mode is allowed in the other cases.

4.2.6 SetTask()

This function sets the following variables: start and end times $(\tau^s_{\bar{j}}, \tau^e_{\bar{j}})$ of current job $\bar{j}$ in the current time shift $\bar{\tau}$; at each iteration of the heuristic, the following parameters are defined and updated:

– The first available time of each machine m: τ_{a_m};
– The first available time of each worker w: τ_{a_w};
– The first available time of each job j: τ_{a_j}; for example, at time zero, this value can be the release date of the job.
– The residual work of $\bar{j}$, $r_{\bar{j}}$; at time 0, $r_{\bar{j}} = p_{\bar{j}}$ and afterwards its value is updated after each assignment, $r_{\bar{j}} = r_{\bar{j}} - (\tau^e_{\bar{j}} - \tau^s_{\bar{j}})$.

Different scenarios may occur according to the value returned by function "EvaluateConduction()" and to the current state of the different involved factors: machine, job, and worker.

Scenario 1 $EvaluateConduction(\bar{w}) = 2$. In this case, the job $\bar{j}$ can start at the first instant time at which it is ready to be executed and all required resources are available: $\tau^s_{\bar{j}} = \max(\bar{\tau}^s, \tau_{a_m}, \tau_{a_{\bar{w}}}, \tau_{a_{\bar{j}}})$. If $\tau^s_{\bar{j}} + r_{\bar{j}} <= \bar{\tau}^e$, the current job can end its processing during the current shift, as we can see in Fig. 1. Hence, $\tau^e_{\bar{j}} = \tau^s_{\bar{j}} + r_{\bar{j}}$ and the other parameters get updated in this way:

– $\tau_{a_m} = \tau_{a_{\bar{w}}} = \tau_{a_{\bar{j}}} = \tau^e_{\bar{j}}$;
– $r_{\bar{j}} = 0$;

if $\tau^s_{\bar{j}} + r_{\bar{j}} > \bar{\tau}^e$, $\tau^e_{\bar{j}} = \bar{\tau}^e$ and $r_{\bar{j}} = r_{\bar{j}} - (\tau^e_{\bar{j}} - \tau^s_{\bar{j}})$. The generic function used to calculate the end processing time of a job is given by $\tau^e_{\bar{j}} = \min(\tau^s_{\bar{j}} + r_{\bar{j}}, \bar{\tau}^e)$.

Scenario 2 $EvaluateConduction(\bar{w}) = 1$. This condition occurs when the worker is already busy on another machine, but parallel working mode is not allowed. In order to calculate the first available starting time of the job $\bar{j}$, it is also necessary to take into account the worker state on the other machine, as shown in Fig. 2. In fact, in this case, $\tau^s_{\bar{j}} = \max(\bar{\tau}^s, \tau_{a_m}, \tau^e_i)$ and $\tau^e_{\bar{j}} = \min(\bar{\tau}^e, (\tau^s_{\bar{j}} + r_{\bar{j}}))$.
The other parameters get updated in this way:

– $\tau_{a_m} = \tau_{a_{\bar{j}}} = \tau^e_{\bar{j}}$;
– $\tau_{a_{\bar{w}}} = \min(t^e_{\bar{j}}, \tau^e_i)$;
– $r_{\bar{j}} = \max(0, (\tau^s_{\bar{j}} + r_{\bar{j}}) - \bar{\tau}^e)$;

4.2.7 EnQueue()

This function is used only if the residual work of the current job $\bar{j}$ is greater than zero. In this case, the job is pushed back in the queue of machine $\bar{m}$. Since the queue $Q_{\bar{m}}$ is a *LIFO* queue, the job $\bar{j}$ will be the first extracted by the function $DeQueue(\bar{m})$ when the machine $\bar{m}$ will be addressed in a successive iteration.

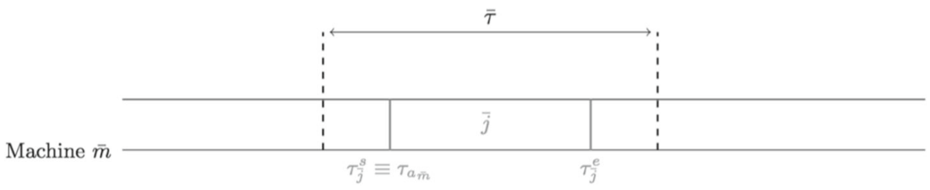

Fig. 1 Scenario 1

5 Computational results

The proposed approach was tested on various real-world instances of the problem, collected during a six months observation period in the partner firm. The targeted shop floor includes up to 17 work centers, supervised by an overall team of up to 19 workers, whose presence is spread over a two or three shifts per day rotation scheme. No worker can be assigned to more than a single shift per day. Consequently, the number of available workers is always lower than the number of machines in each shift, with typical values between 5 and 10 workers per shift.

The first problem solved by the proposed approach assigns and schedules the operations on the available machines, considering the resulting sequence-dependent setup times and ignoring the workers availability, with the objective to minimize the makespan and the cumulated start times of all jobs: hence, idle times in the obtained schedule can only be caused, at this level, by machines calendars.

Taking into account the availability of the workers, with all the specific issues analyzed in Section 3 can, hence, only deteriorate the objective function value. The proposed algorithm, indeed, takes the solution of the CP model, in terms of both the allocation of jobs to machines and of their sequences on the machines (that are not subject to modifications by the algorithm), and allocates workers to machines over time, potentially causing one or more stops on each machine, due to the possible lack of a human supervision (given the founding hypothesis of the work that there is an excess of machines compared with the available human resources), but can never, by construction, improve the starting CP solution. In other words, the result obtained with the first part of the proposed approach provides a valid lower bound to the objective function achievable when all the aspects of the problem are considered.

On the other hand, not considering the aspects that can improve the system performances, namely the possibility for workers to supervise more than a machine, is a way to define an upper bound for the objective function.

Since the literature does not provide any study that addresses such a specific problem, considering both the lower and the upper bounds allow to calculate a range of values that can be used to evaluate the quality of the results obtained with the proposed approach.

During the testing period, the firm investigated four different configurations of the shop floor, each mainly characterized by a different availability of workforce over the shifts and by the choice to activate or not a couple of machines, that can also be used by another division in the plant. The four configurations are reported in Table 3, where:

- M is the number of machines available in the considered shop floor;
- T_1 is the number of workers available during the first shift;
- T_2 is the number of workers available during the second shift;
- T_3 is the number of workers available during the third shift.

The results are presented in Table 4, where:

- *Id* is the unique identifier of the test;
- *Config* is the configuration of the shop floor for the test (one of the four configurations reported in Table 3);

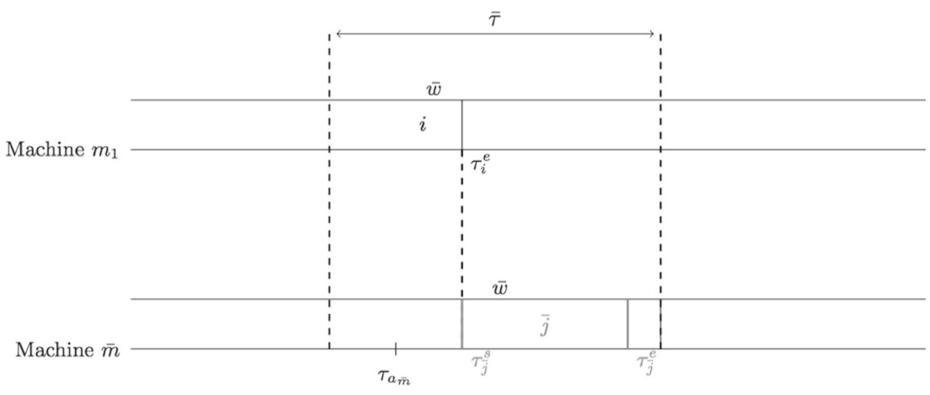

Fig. 2 Scenario 2

Table 3 Configurations

Config.	M	T_1	T_2	T_3
A	15	10	8	0
B	17	6	6	0
C	15	6	6	6
D	17	7	6	6

- *LB (Lower Bound)* is the lower bound provided by the CP model solution;
- *OBJ* is the objective value of the proposed approach, i.e., CP model plus Constructive Heuristic;
- *UB (Upper Bound)* is the objective value of the hierarchical approach, where parallel working modes for workers are not allowed;
- Δ is the difference (in working days and fraction of a day) between the upper and lower bounds, $\Delta = UB - LB$, providing a reference range between the hypothetical lower bound and the upper bound that does not exploit parallelism;
- Δ_{LB} is the difference (in working days and fraction of a day) between the objective value and the lower bound, $\Delta_{LB} = OBJ - LB$, representing an estimation of the "cost" for having less workers than machines;
- Δ_{UB} is the difference (in working days and fraction of a day) between the upper bound and the objective value, $\Delta_{UB} = UB - OBJ$, providing an estimation of the benefit provided by the usage of parallelism;
- $\%\Delta_{LB}$ is the percentage value of the ratio $\frac{\Delta_{LB}}{\Delta}$;
- $\%\Delta_{UB}$ is the percentage value of the ratio $\frac{\Delta_{UB}}{\Delta}$;
- *Days* is the number of working days in the schedule;
- *%Gap* is the optimality gap in percent, calculated as $\frac{\Delta_{LB}}{Days}$, representing an estimation of the theoretically possible further improvement of the objective if the number of workers is increased enough to allow the continuous operation of all the available machines.

A test was performed each time the plant manager needed a production plan for the forthcoming period, which did not happen with an exact frequency, but was related to the availability of new customers orders. The twenty tests reported in Table 4 cover all the planning events that took place in the six months horizon considered for the validation of the proposed method. The tests were performed with a forced 2-h time limit on a stand-alone PC dedicated to the tests, with the following configuration: 2.8GHz Intel i7$^{\text{TM}}$ quad-core CPU, 16GB RAM, 1TB SSD Hard Drive, Microsoft Windows$^{\text{TM}}$ 10 Pro 64bit, IBM ILOG$^{\text{TM}}$ Optimization Studio 12.10.

The results obtained show how, for config A cases, and in part also for config C cases, the proposed heuristic is able to

Table 4 Result table

Id	Config.	LB	OBJ	UB	Δ	Δ_{LB}	Δ_{UB}	$\%\Delta_{LB}$	$\%\Delta_{UB}$	Days	$\%GAP$
SA1	A	23/08/19 08:05	23/08/19 15:15	26/08/19 08:58	1.04	0.30	0.74	29%	71%	9.39	3%
SA2	A	29/08/19 22:18	30/08/19 11:37	02/09/19 07:28	1.38	0.55	0.83	40%	60%	16.23	3%
SA3	A	03/09/19 11:20	04/09/19 06:13	05/09/19 06:10	1.78	0.79	1.00	44%	56%	16.01	5%
SA4	A	17/09/19 13:45	17/09/19 21:13	18/09/19 17:04	1.14	0.31	0.83	27%	73%	12.63	2%
SA5	A	02/10/19 12:11	03/10/19 13:28	04/10/19 06:11	1.75	1.05	0.70	60%	40%	16.31	6%
SB1	B	17/10/19 13:31	23/10/19 08:16	24/10/19 21:11	5.32	3.78	1.54	71%	29%	18.09	21%
SB2	B	22/10/19 07:41	29/10/19 12:46	30/10/19 16:02	6.35	5.21	1.14	82%	18%	16.95	31%
SB3	B	28/10/19 21:41	05/11/19 09:52	05/11/19 19:37	4.91	4.51	0.41	92%	8%	15.16	30%
SB4	B	11/11/19 13:43	15/11/19 19:43	18/11/19 19:36	5.25	4.25	1.00	81%	19%	15.57	27%
SB5	B	21/11/19 15:41	26/11/19 13:26	28/11/19 08:02	4.68	2.91	1.78	62%	38%	13.31	22%
SC1	C	28/11/19 04:14	29/11/19 15:51	30/11/19 02:47	1.94	1.48	0.46	77%	23%	8.41	18%
SC2	C	13/12/19 18:34	16/12/19 13:07	17/12/19 21:27	2.12	0.77	1.35	36%	64%	12.30	6%
SC3	C	19/12/19 14:34	21/12/19 01:07	24/12/19 05:05	2.60	1.44	3.17	55%	45%	14.80	10%
SC4	C	13/01/20 10:09	15/01/20 06:16	17/01/20 21:42	2.48	1.84	2.64	74%	26%	15.01	12%
SC5	C	24/01/20 14:32	28/01/20 17:39	29/01/20 16:11	3.07	2.13	0.94	69%	31%	13.49	16%
SD1	D	31/01/20 09:52	05/02/20 02:00	05/02/20 15:10	3.22	2.67	0.55	83%	17%	10.83	25%
SD2	D	10/02/20 06:43	11/02/20 23:42	13/02/20 05:38	2.95	1.71	1.25	58%	42%	12.74	13%
SD3	D	18/02/20 15:20	20/02/20 14:50	21/02/20 20:14	3.20	1.98	1.22	62%	38%	13.37	15%
SD4	D	28/02/20 07:55	03/03/20 09:15	04/03/20 13:51	3.25	2.06	1.19	63%	37%	11.55	21%
SD5	D	07/03/20 05:44	12/03/20 00:12	13/03/20 01:24	3.82	2.77	1.05	73%	27%	13.76	20%

come close to the ideal results of the lower bound. A higher concentration of workers per shift, indeed, allows to very closely match the ideal condition, in which all the machines are permanently supervised, and hence operational, during each working shift. Config A and, to a lesser extent, config C cases are characterized by a more "balanced" ratio between the available machines and the workers assigned to each shift: in these situations, the proposed approach shows its effectiveness, almost achieving the ideal results even if there are less workers than machines, because it fully exploits the parallel working modes. On the other hand, the more "unbalanced" availability of workers in config B and config D cases results in a higher distance between the solution and the lower bound, though remaining within a comparable distance in terms of overall gap.

6 Conclusions

This work takes its inspiration from an industrial case where the production planning of a FFSP with additional workforce-related constraints needs to be considered. The proposed approach iterates over the stages of the problem, corresponding to the shop floors in the plant, addressing each stage with an ad hoc procedure based on constraint programming and a constructive heuristic. The approach is validated on real test cases collected over several months of experimentation and the achieved results show its effectiveness to solve real instances of the problem.

Future work will investigate the possibility to improve the second phase of the single-stage solution technique, evaluating the possible improvements achievable through a more sophisticated metaheuristic approach or considering a constraint programming formulation.

The problems solved and presented in Section 5 are similar in terms of number of jobs and available resources: another research topic will be the generation of random but realistic test instances to be able to benchmark the behavior of the proposed method on instances that significantly vary in dimension.

The data used for the computational results are instances of real production planning problems and cannot be made publicly available, since they belong to the firm providing the test case for the work. However, one of the authors is the reference contact for the firm and can be reached for specific requests.

Acknowledgments The paper describes the research activities developed in the experimental case *Prosino S.r.l.* The data for the computational tests have been extracted by the actual ERP (SAP). The authors would like to thank Altea Federation (A. Ruscica, P. Violini, S. Solano, F. Agazzone, A. Lopatriello) and Prosino S.r.l. for the contribution to the experimental campaign.

Funding Open access funding provided by Universit del Salento within the CRUI-CARE Agreement.

References

1. Pinedo M (2012) Scheduling, vol 5. Springer, Berlin
2. Wang H (2005) Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions. Expert Syst 22(2):78–85
3. Benkalai I, Rebaine D, Baptiste P (2019) Scheduling flow shops with operators. Int J Prod Res 57(2):338–356
4. Gong G, Chiong R, Deng Q, Han W, Zhang L, Lin W, Li K (2020) Energy-efficient flexible flow shop scheduling with worker flexibility. Expert Syst Appl 141:112902
5. Słowiński R. (1980) Two approaches to problems of resource allocation among project activities—a comparative study. Journal of the Operational Research Society 31(8):711–723
6. Edis EB, Oguz C, Ozkarahan I. (2013) Parallel machine scheduling with additional resources: notation, classification, models and solution methods. Eur J Oper Res, pp 449–463
7. Edis EB, Oguz C (2012) Parallel machine scheduling with flexible resources. Computers & Industrial Engineering 63(2):433–447
8. Daniels RL, Hoopes BJ, Mazzola JB (1996) Scheduling parallel manufacturing cells with resource flexibility. Management Science 42(9):1260–1276
9. Zouba M, Baptiste P, Rebaine D (2009) Scheduling identical parallel machines and operators within a period based changing mode. Computers & Operations Research 36(12):3231–3239
10. Chen Z-L (2004) Simultaneous job scheduling and resource allocation on parallel machines. Ann Oper Res 129(1-4):135–153
11. Hu P-C (2005) Minimizing total flow time for the worker assignment scheduling problem in the identical parallel-machine models. Int J Adv Manuf Technol 25(9-10):1046–1052
12. Artigues C, Gendreau M, Rousseau L-M (2006) A flexible model and a hybrid exact method for integrated employee timetabling and production scheduling. In: International Conference on the Practice and Theory of Automated Timetabling. Springer, pp 67–84
13. Kellerer H, Strusevich VA (2003) Scheduling problems for parallel dedicated machines under multiple resource constraints. Discret Appl Math 133(1-3):45–68
14. Costa A, Cappadonna FA, Fichera S (2013) A hybrid genetic algorithm for job sequencing and worker allocation in parallel unrelated machines with sequence-dependent setup times. Int J Adv Manuf Technol 69(9-12):2799–2817

15. Afzalirad M, Rezaeian J (2016) Resource-constrained unrelated parallel machine scheduling problem with sequence dependent setup times, precedence constraints and machine eligibility restrictions. Computers & Industrial Engineering 98:40–52
16. Afzalirad M, Shafipour M (2018) Design of an efficient genetic algorithm for resource-constrained unrelated parallel machine scheduling problem with machine eligibility restrictions. J Intell Manuf 29(2):423–437
17. Al-harkan IM, Qamhan AA (2019) Optimize unrelated parallel machines scheduling problems with multiple limited additional resources, sequence dependent setup times and release date constraints. IEEE Access 7:171533–171547
18. Abdolshah M (2014) A review of resource-constrained project scheduling problems (RCPSP) approaches and solutions. International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies 5(4):253–286
19. Demeulemeester EL, Herroelen WS (2006) Project scheduling: a research handbook, vol 49. Springer Science & Business Media, Berlin
20. Dorndorf U (2002) Project scheduling with time windows: from theory to applications; with 17 tables. Springer Science & Business Media
21. Neumann K, Schwindt C, Zimmermann J (2012) Project scheduling with time windows and scarce resources: temporal and resource-constrained project scheduling with regular and nonregular objective functions. Springer Science & Business Media
22. Kis T (2005) Project scheduling: a review of recent books. Oper Res Lett 33(1):105–110
23. Blazewicz J, Lenstra JK, Rinnooy Kan AHG (1983) Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics 5(1):11–24
24. Hooker J (2011) Logic-based methods for optimization: combining optimization and constraint satisfaction, vol 2. Wiley, Hoboken
25. Van Hentenryck P (1999) The OPL optimization programming language. MIT Press, Cambridge
26. IBM ILOG (2018) IBM ILOG CPLEX Optimization Studio v12.8.0 documentation
27. Laborie P, Rogerie J, Shaw P, Vilím P (2018) IBM ILOG CP optimizer for scheduling. Constraints 23(2):210–250