A Multi-Agent Framework to Solve Energy-Aware Unrelated Parallel Machine Scheduling Problems with Machine-Dependent Energy Consumption and Sequence-Dependent Setup Time

Giancarlo Nicolò,[1,2] **Sergio Ferrer,**[1,2] **Miguel A. Salido,**[1,2] **Adriana Giret,**[2] **Federico Barber**[1,2]
[1]Instituto de Automática e Informática Industrial, Valencia, Spain
[2]Universitat Politècnica de València, Valencia, Spain
giani1@dsic.upv.es, serfers2@dsic.upv.es, msalido@dsic.upv.es, agiret@dsic.upv.es, fbarber@dsic.upv.es

Abstract

Nowadays,the manufacturing industry consumes almost half of the energy produced on the planet. Within this industry, sustainable scheduling represents a main issue due to the energy efficiency that it is able to achieve. Centralized or decentralized approaches to sustainable scheduling can be found in state-of- the-art techniques using both optimal or heuristic methods. In this paper, a distributed system modelled as a multi-agent framework is proposed to solve energy-aware scheduling problems by applying it to a real case of the injection air moulding industry, which has intense energy consumption. In this proposal, a set of agents collaborate to reach an agreement in order to minimize several objective functions based on: total job weighted tardiness, total setup time, and energy consumption. The proposed system is compared to centralized and distributed approaches, highlighting advantages and drawbacks and providing an interesting point of view about how to tackle energy-aware scheduling with new agreement techniques in a distributed environment.

1 Introduction

In recent years, the growth of sustainable industry has been increasingly encouraged. Nowadays, one of the aspects that is gaining strength in this area is the study of energy efficiency in the manufacturing industry (Seow and Rahimifard 2011; Tonelli, Evans, and Taticchi 2013; Liu, Yang, and Cheng 2017). Practically half of the energy consumed on the planet is used by the industrial sector (Newman et al. 2012), and energy-aware scheduling problems are considered to be a fundamental issue in improving the efficiency of energy inputs in the industrial environment (Bruzzone et al. 2012; Dai et al. 2013; Plitsos et al. 2017). For this reason, there is a growing need to perform multi-objective optimizations in scheduling problems, which traditionally try to minimize only measures related to process time, so that they also include among their priorities achieving energy efficient solutions for sustainability purposes. Centralized or decentralized approaches can be found in the state-of-the-art literature. Although most approaches found in the state of the art can be classified as centralized, a few decentralized proposals that tackle scheduling problems can be found. (Agnetis et al. 2014) presents a multi-agent approach in which subsets of jobs that share the same resources are evaluated with different criteria for scheduling on one or more machines (shared resources). In (Liu, Abdelrahman, and Ramaswamy 2007), a multi-agent framework is proposed for dynamic job scheduling. This approach finds solutions by applying solvers that are distributed in multiple agents to fix jobs in a dynamic environment knowing minimal information about the global state. (Martin et al. 2016) proposes a distributed system based on agents, where each agent implements a different metaheuristic or local search and cooperates with the other agents through peer-to-peer messages. However, each agent works on the whole problem. (Behnamian 2016) studies a scheduling problem in a multi-factory environment, where he presents a distributed and parallel approach exploiting a discrete particle swarm optimization algorithm. In (Han et al. 2017), the authors present a system framework with integrated multi-agent technology using a mathematical model that is proposed for solving a two-stage scheduling problem. In (Tonelli et al. 2016), a simple multi-agent system model is proposed to decompose an energy-aware scheduling problem into smaller subproblems. In this approach, each agent solves a subproblem by using a Mixed Integer Linear Programming (MILP) model and the results are combined to obtain a global solution. A similar idea is proposed in (Nicolò et al. 2017) where a set of solving techniques are compared and job features are studied in order to tackle energy-aware scheduling problems. Taking into account the proposals and results from (Tonelli et al. 2016; Nicolò et al. 2017), in this paper, a multi-agent framework is presented for solving energy-aware scheduling problems where every task can be executed on a different set of machines with different processing times and energy consumption on each machine. This framework uses a distributed structure for allocating different agents that execute simultaneous local searches on previously found solutions. To validate the proposed system, it is applied in a real industrial case with intensive energy consumption. Specifically, we applied it to one of the largest industries in the large-scale production sector, plastic production by air injection moulding, which is among the industries that consume the most energy ($2.06 \cdot 10^8$ GJ per year only in the USA). The problem that we work with consists of scheduling a set of job orders (or tasks) in a set of parallel moulding injection presses, where each order is characterized by a kind of product to be produced and on which a penalty cost is to be paid if the task finishes after a deadline. A set of alternative presses is available for each job where both the energy consumption and the time needed to finish the job directly depend on the machine selected. Every time a task is finished, it is necessary to carry out cleaning tasks and to change the mould for the next job order. Therefore, a setup time between different tasks must be taken into account. The setup time depends on the pair of jobs that are executed sequentially and the machine that is used for the execution of those jobs. Thus, the presented problem can be interpreted as a multi-objective scheduling problem where the main goal is to minimize three different factors: total tardiness, total energy consumption, and total setup time between jobs (Lu et al. 2012; Gong et al. 2017). This problem presents a structure that is similar to other optimization issues in the manufacturing industry consisting of the scheduling of independent jobs on a set of unrelated parallel machines with different setup or execution times, depending on the machine (Manupati et al. 2017; Che, Zhang, and Wu 2017). Problems of this kind are well-known cases of computation intractable problems (Allahverdi 2015). It is worth noting how the simplification of the analyzed scenario (the single machine total weighted tardiness problem) belongs to NP-hard problems (Lawler 1977), outlining the intrinsic difficulty of the problem under consideration. In this paper, a distributed approach that includes an internal agreement system among agents is presented as an alternative strategy to tackle multi-objective scheduling problems. Additionally, the proposed structure is independent from the search strategy used, so it can be used to implement different search techniques as presented in the further sections below. The evaluation section presents a comparison between the different techniques implemented in our system and a centralized approach that models the problem as a mixed integer linear programming in (Paolucci, Anghinolfi, and Tonelli 2015). Moreover, a comparison with a multi-objective-based Genetic Algorithm and with the MILP results obtained in (Tonelli et al. 2016) is carried out in order to evaluate the gains of including collaborative reasoning in the search process.

2 Description of the Scheduling Problem

The problem to be solved consists of scheduling a set of orders (each of which corresponds to a job) on a set of unrelated parallel machines (injection air molding presses). Each job has two temporal features associated to it: a release date, which is the earliest moment to start processing the job; and a due date, which indicates the latest instant of time at which the job has to be finished. When a job is delayed with respect to its due date, for each unit of time exceeding the expected finishing time, there is a penalty cost expressing the job priority. In other words, a delayed job with high priority will have greater penalties when it is than a job with a lower priority. Each job must be processed on one machine selected from among a set of eligible machines that are specific for each job. Jobs that can be executed on more than one machine are called *shared jobs* (i.e., the set of eligible machines for these jobs has more than one element). The energy consumption and processing time of each job depend on the machine selected to execute it. This means that different machines may need different amounts of energy and time to execute the same job; for example, a new machine can be faster and energetically more efficient than an older one. Every time a job finishes its execution, some physical actions must be carried out on the machine before it can process another job: the machinery must be cleaned, the mould must be changed, etc. Therefore, a setup time between jobs must be taken into account. The setup time directly depends on the pair of jobs that are executed sequentially and also on the features and particularities of the machine. Therefore there is a different setup time for each pair of jobs on each machine. In the next section, the problem under observation is formally described and its mathematical formulation is presented in a Mixed Integer Linear Programming model.

2.1 Mathematical formulation

From the problem description given in the introduction, the following scheduling notation (Graham et al. 1979) is presented:

$$R_m|M_j, p_{jk}, E_{jk}, r_j, s_{ijk}| \sum w_j T_j, \sum E_{jk}, \sum S_{ijk}$$

The problem is multi-objective since there are three measures to be minimized, which are expressed as objective functions: the total weighted tardiness of the jobs $TT(s)$, the total energy consumption $EN(s)$, and the total setup time $ST(s)$. The solution s^* can be obtained by minimizing a 3-dimensional objective function:

$$s^* = \arg\min_{s \in S}[TT(s), EN(s), ST(s)] \qquad (1)$$

where S denotes the feasibility space for the problem solution space. To represent the problem model and the objective function components to be optimized, a list of notations extracted from (Paolucci, Anghinolfi, and Tonelli 2015) is presented below.

Sets:

- $J = \{1, \dots, n\}$, the set of jobs, indexes 0 and $n+1$ denote two fictitious jobs corresponding to the first and last job on each machine
- $M = \{1, \dots, m\}$, the set of machines (i.e., the presses)
- $M_j, \forall j \in J$, the set of machines that can execute job j
- $J_k, \forall k \in M$, the set of jobs that can be executed by machine k

Parameters:

- B, a sufficiently large constant
- $D_j, \forall j \in J$, the due date of job j
- $R_j, \forall j \in J$, the release date of job j
- $W_j, \forall j \in J$, the tardiness penalty of job j
- $P_{jk}, \forall j \in J, \forall k \in M_j$, the processing time of job j on the eligible machine k
- $E_{jk}, \forall j \in J, \forall k \in M_j$, the energy consumption for processing job j on the eligible machine k
- $S_{ijk}, \forall i, j \in J, \forall k \in M_j \cap M_i$, $i \neq j$, the setup time on machine k between the completion of job i and the start of the subsequent job j
- $\Pi_g, g = 1, 2, 3$, the weights of the objective function components (i.e., total weighted tardiness, total energy consumption, and total setup time)

Variables:
- $c_j, \forall j \in J \cup \{0\}$, the completion time of job j
- $t_j, \forall j \in J$, tardiness of job j with respect to its due date
- $x_{ijk} \in \{0,1\}, \forall i,j \in J \cup \{0, n+1\}, k \in M_i \cap M_j, i \neq j$, binary sequencing variables (i.e., $x_{ijk} = 1$ denotes that job i immediately precedes job j on machine k)
- $y_{jk} \in \{0,1\}, \forall j \in J, k \in M_j$, binary assignment variables (i.e., $y_{jk} = 1$ denotes that j is processed by k)

2.2 Mixed integer programming model

In order to being able to evaluate a solution of the problem, the three objective functions must be aggregated. Furthermore, the three factors to be combined have different dimensions (energy and time), and the conversion to a common dimension may not always be practical because decision makers may have difficulty expressing preference information through numerical weights, taking into account the original dimension of the objective function components. The mixed integer linear programming (MILP) model (Paolucci, Anghinolfi, and Tonelli 2015) combines the three factors into a scalar function with a minimum deviation method, resulting in the following scalar objective function F to be minimized:

$$\min\ \Pi_1 \cdot \frac{\sum_{j \in J} W_j \cdot t_j - f_1^-}{f_1^+ - f_1^-} + \Pi_2 \cdot \frac{\sum_{j \in J} \sum_{k \in M_j} E_{jk} \sum_{\substack{i \in J_k \\ i \neq j}} x_{ijk} - f_2^-}{f_2^+ - f_2^-} + \Pi_3 \cdot \frac{\sum_{k \in M} \sum_{i \in J_k} \sum_{\substack{j \in J_k \\ i \neq j}} S_{ijk} \cdot x_{ijk} - f_3^-}{f_3^+ - f_3^-} \tag{2}$$

The quantity f_g^-, g $\in \{1,2,3\}$ in (2) represents the best (i.e., minimum) value for the g-th component when this is optimized individually; f_g^+ is an estimation of the worst value for $f_g(s)$ that can be fixed as $f_g^+ = \max_{h \neq g} f_g(s_h^*)$, where (s_h^*) is the optimal solution found when the objective $f_h(s)$ is individually optimized. The weights Π_g, $g \in \{1,2,3\}$ in (2) express the relative importance given by the decision maker to the different objective components and are selected such that $\sum_g \Pi_g = 1$. The above function (2) is subject to:

$$\sum_{\substack{i \in J_k \\ i \neq j}} x_{ijk} = y_{jk} \qquad \forall j \in J, k \in M_j \tag{3}$$

$$\sum_{\substack{j \in J_k \\ j \neq i}} x_{ijk} = y_{ik} \qquad \forall i \in J, k \in M_i \tag{4}$$

$$\sum_{k \in J_k} y_{jk} = 1 \qquad \forall j \in J \tag{5}$$

$$\sum_{j \in J_k} x_{0jk} \leq 1 \qquad \forall k \in M \tag{6}$$

$$c_j \geq R_j + \sum_{k \in M_j} P_{jk} y_{jk} \qquad \forall j \in J \tag{7}$$

$$t_j \geq c_j - D_j \qquad \forall j \in J \tag{8}$$

$$c_j \geq c_i + P_{jk} + S_{ijk} - B \cdot (1 - x_{ijk}) \qquad \forall k \in M, \forall i,j \in J_k, i \neq j \tag{9}$$

$$c_0 = 0 \tag{10}$$

$$c_j \geq 0, t_j \geq 0 \tag{11}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i,j \in J, i \neq j, k \in M_i \cap M_j, \tag{12}$$

$$y_{jk} \in \{0,1\} \qquad \forall j \in J, k \in M_j \tag{13}$$

Constraints (3) and (4) impose that each job assigned to a machine must be sequenced on that machine. Specifically, it must have a predecessor and a successor on the machine. Constraint (5) guarantees that each job is assigned to a single machine among the ones eligible to process it. Constraint (6) imposes that, at most, a single job is the first one scheduled on each machine. Constraint (7) defines the lower bound for the job completion time, and Constraint (8) defines the job tardiness. Constraint (9) controls the job completion times, ensuring that each machine processes one job at a time and the setup time between two successive jobs is satisfied. Constraint (10) fixes(sets) the completion time for the dummy job_0, and Constraints (11), (12) and (13) define the problem decision variables.

3 System Proposal

To find a solution to the problem introduced, a multi-agent framework is developed as a decentralized solution to scale up the search process and to reduce the computational burden derived from the combinatorial explosion of the problem (Lawler 1977). The motivation to introduce a distributed framework comes from the observation that the problem under analysis can be optimally decomposed if each job could only be executed on one machine (i.e., $\forall j \in J : |M_j| = 1$). In this scenario, the search process can be optimally distributed over as many independent scheduling problems as the number of machines m, achieving an optimal problem decomposition in m independent *single machine scheduling subproblems*, where each one of them is still *NP-Hard*. Unfortunately, most jobs can be executed on different machines (i.e., shared-jobs). Therefore, another source of combinatorial explosion is how to optimally assign each shared job to an appropriate machine in order to optimize the given objective function, namely the *job assignation subproblem*. Based on the above considerations, a Multi-Agent framework is proposed to tackle the computational burden of both subproblems through an iterative process that is composed of agreements and distributed searches. The agreements are meant to address the job assignation subproblem through a search space reduction strategy that restricts the set of eligible machines for a specific shared job, while the distributed searches solve the independent machine scheduling subproblems that are obtained from the restricted sets. The rest of this section presents a brief overview of the proposed framework, while a formal description of its components is reported in section 4. Before moving on to the framework overview, definitions 1 to 9 are reported to establish uniformity of notation.

Definition 1. *A* shared-job j *is a job where the set of eligible machines contains more than one element, formally:* $j \in J$ *is a* shared-job $\leftrightarrow |M_j| > 1$.

Definition 2. *A* job assignment Y^*_{jw} *is the assignation of a specific job j to an eligible machine w, formally: $Y^*_{jw} = \{\, y_{jw} \bigcup_{w \neq w'} y_{jw'} \mid w, w' \in M_j \}$, where y_{jk} are the problem decision variables with $y_{jw} = 1 \wedge y_{jw'} = 0$.*

Definition 3. *A* complete job assignation Y^* *is the union set of all job assignments for the problem, including both shared and non-shared job, formally: $Y^* = \{\, \bigcup Y^*_{jw} \mid j \in J \,\}$.*

Definition 4. *A* restricted set of eligible machines for a shared job *j, M^*_j is any subset of the initial set of eligible machines for the shared-job j, formally: $M^*_j \subset M_j$. Without loss of generality, the same definition can be extended to the case of a set of shared-jobs J'.*

Definition 5. *A* restriction over the set of eligible machines for a shared-job *j $r(M_j, w)$ is a function returning the restricted set of eligible machines for j such that machine w is removed, formally: $r(M_j, w) = M_j \setminus w$. Without loss of generality, the same definition can be extended to the case of a set of shared-jobs J'.*

Definition 6. *A* complete job assignement Y^* is coherent over the union of all restricted set of eligible machines: $Y^* \asymp \bigcup M^*_j$ *if and only if, for each job assignment Y^*_{jw} in Y^*, the machine w assigned to job j is included in the restricted set of eligible machines for j. Formally: $Y^* \asymp \bigcup M^*_j \Leftrightarrow \forall Y^*_{jw} \in Y^* : w \in M^*_j$.*

Definition 7. A partial problem instance over a specific machine *w, $DataSet(w)$ is the projection of the problem instance to machine w, formally: $DataSet(w) = \{\, J^*_w,\ D_j,\ R_j,\ W_j,\ P_{jw},\ E_{jw},\ S_{ijw},\ \Pi_g \mid i, j \in J^*_w \,\}$, where J^*_w is the set of jobs assigned to machine w.*

Definition 8. *A* local solution s_w *is the solution of the partial problem instance on a specific machine w, formally: $s_w = \{\, \bigcup y_{jw} \bigcup x_{ijw} \bigcup c_j \bigcup t_j \mid i, j \in J_w \}$.*

Definition 9. *A* job property *is a binary relation $\mathcal{R}$ between a job j and its local or global schedule solution s. It is defined by evaluating the job informations extracted from the analysis of the job over the schedule (e.g., job position, most energy consuming job, etc.).*

3.1 Overview of the system

The proposed work is described as a multi-agent framework rather than a usual system (Tonelli et al. 2016) in order to create an abstraction that represents a general view of how to confront the problem under analysis. Specifically, the framework is composed of different entities, namely agents, that are described in terms of their functionalities so that specific interactions between them are meant to achieve/implement a macro-functionality that we call a framework *module*. In order to express the framework generality with respect to the problem under analysis, both the agent functionalities and modules are described in terms of their specifications rather than concrete implementations. In this scenario, the concrete implementation has the role of tuning the developed system to fulfill its non-functional requirements (e.g., bounded computational time, reactiveness over problem perturbation, etc ...). To visualize the above idea, Figure 1 shows the search process workflow of a problem instance solution. Specifically, there are two different type of agents $A = \{A_s \cup A_w\}$: *supervisor* ($s \in A_s$) and *worker* ($w_i \in A_w = \{1, \ldots, m\}$). Each module is represented by a rectangle that is included in a larger figure with shaded lines indicating the group of agents that is in charge of implementing the module. The solution search process starts with the preprocessing of problem instance information via the *Initializer* module, which is implemented by the supervisor. The main module's objectives are to analyze the data in order to identify shared jobs and to construct related sets of eligible machines, while instantating as many worker agents as machines in the problem. After the initialization phase, the machine assignation subproblem is tackled through the *Centralized Solver* module implemented by the supervisor. Its objective is to produce a complete job assignation Y^* and an optional initial solution (referred as *global solution* s^I) where both of them have to be coherent with the union of currently agreed restricted sets of eligible machines $\bigcup M^*_j$. The outputs produced by the centralized solver are sent to the macro module *Collaborative Reasoner*, which is composed of submodules: *Distributed Search* and *Agreement*, both of which are implemented by worker agents. The goal of this module is to solve the single machine scheduling subproblems in a distributed way. The first submodule receives the job assignation and starts a distributed search to produce local solutions s_{w_i} with related local job properties of the shared jobs. If an initial global solution s^I is received from the previous module, the searching process has to guarantee the same or higher solution quality as the initial one ($F(\bigcup s_w) \leq F(s^I)$). The solutions s_{w_i} with related shared-job properties are then used in the *Agreement* module, whose main goal is to find an initial agreement over a shared-job j' and a set of restrictions $M^*_{j'}$. After the initial agreements, a reschedule procedure is invoked as a particular case of the distributed search, whose aim is to validate $M^*_{j'}$ through the analysis of a distributed

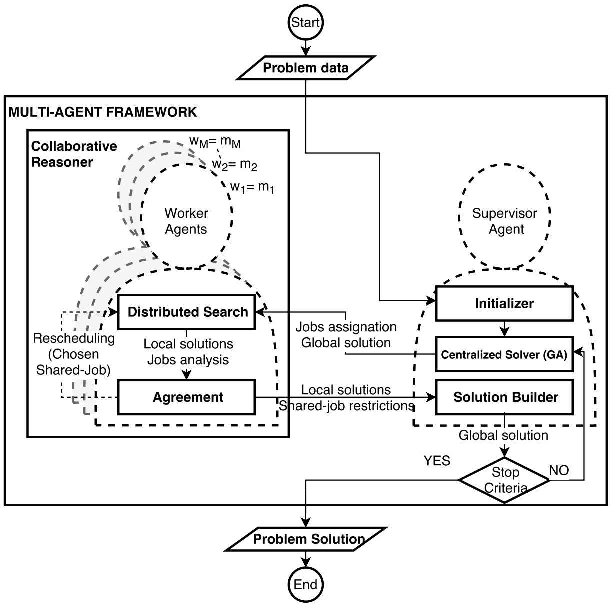

Figure 1: Framework workflow for the solution search.

rescheduling for the agreed job j' over all its current eligible machines. Once the $M^*_{j'}$ are validated for local solutions s_{w_i}, the agreement terminates and the set of restrictions will be fixed, meaning that during the next iterations of the workflow, the machines removed from the eligible set won't ever be analyzed for the agreed job. The outcome produced by the collaborative reasoner module is shared by the worker agents with the supervisor in order to implement the last module of the framework: *Solution Builder*. The goal of the module is to collect s_{w_i} and validate $\bigcup M^*_{j'}$ to build a final global solution s^F. Moreover, $\bigcup M^*_{j'}$ is propagated to the centralized solver module in order to produce valid job assignations for the next iterations (e.g., after the machine w' is removed from the eligible set of the chosen job j', the centralized solver does not have to produce any job assignment of j' to w'). After the final global solution s^F is built, the module checks the fulfilment of the stop criteria. If the stop criteria is satisfied, the framework returns s^F otherwise, a new iteration of the framework is performed.

4 System Modules

This section presents an extensive description of the framework components presented above. Each module description starts by introducing abstract specifications and then reporting the concrete implementation (with specific design choices) that is used to build the systems for the evaluation. The following subsections describe each of the modules, introducing them in the same order as the workflow discussed in section 3.1.

4.1 The initializer

The supervisor agent starts the framework through the implementation of the initializer module. This module analyzes the problem data, instantiates the worker agents, and distributes specific information about the problem to them. To do this, the supervisor agent receives the problem instance specification and generates as many worker agents w_i as unrelated parallel machines are involved in the problem, where w_i will simulate machine $k_i \in M$. Then, it creates partial problem instances $DataSet(w_i)$ and assigns them to each worker w_i. To do that, all of the sets of jobs $J^1_{w_i}$ that can only be executed on machine k_i are calculated and used to build the $DataSet(w_i)$. Formally, $J^1_{w_i} = \{j \in J_{k_i} || M_j| = 1\}$ and $DataSet(w_i) = \{\, J^1_{w_i}, D_j, R_j, W_j, P_{jw_i}, E_{jw_i}, S_{j'jw_i}, \Pi_g \,|\, j', j \in J^1_w \,\}$.

4.2 The centralized solver

The centralized solver module is implemented by the supervisor agent with the aim of producing a valid job assignation Y^* and an optional initial solution s^I, both of which have to be coherent with the currently agreed restricted sets of eligible machines $\bigcup M^*_j$. The main goal of this module is to find a set of shared-job assignments that, together with non-shared job assignments, produces a valid job assignation Y^*. No specific functional requirements on Y^* are imposed, which means that any specific approach that satisfies non-functional domain requirements could be used. Moreover, the possibility of producing a global solution s^I that is coherent with Y^* and $\bigcup M^*_j$ is left as optional. Thanks to module generality, any implementation that generates a global solution coherent with $\bigcup M^*_j$ can be used to extract a valid Y^* from the analysis of its decision variables y_{jw}. A job assignation is considered to be valid if it is coherent with the current restrictions on the eligible machine sets that are generated by the collaborative reasoner during previous framework iterations. For our implementation, we used a meta-heuristic solver that implements a genetic algorithm (GA) proposed in (Nicolò et al. 2017).The reason behind this is the fast convergence property of the genetic algorithm and the possibility to reuse the current computation (the produced population) for the next iteration of the workflow, where the supervisor agent is responsible for tuning the genetic algorithm's parameters depending on the non-functional requirements (time constraints) and problem size.

4.3 The collaborative reasoner

The collaborative reasoner macro module is composed of two submodules: *Distributed Search* and *Agreement*, both of which are implemented by worker agents. The goal of the module is to exploit centralized solver outcomes Y^* in order to choose a set of shared-jobs J' and define restricted sets over their eligible machine sets $M^*_{J'}$, while producing local solutions that are coherent with defined restricted sets that have to be fixed for the next framework iterations. This goal is accomplished by exploiting the two submodules: first, a distributed search is performed to find initial local solutions and related job properties, and then an agreement process is executed to select the shared-jobs J'. Thus, a distributed rescheduling of J' is performed to obtain rescheduled local solutions that are analyzed to find an agreement on the restricted sets $M^*_{J'}$. From the above brief description, it can be noted that the number of shared jobs to be chosen and the number of restrictions to be fixed in each iteration are not specified (i.e., remove/fix one or more machines/restrictions from the eligible sef of each job chosen). Moreover, these parameters could be constant or variable during each of the iterations. These design choices are left as implementation choices. For our implementation, we decided to keep both the parameters constant by limiting the number of shared jobs to be chosen per iteration to one, but comparing two choices regarding the number of restrictions to be fixed. In this paper, two opposite values are analyzed: the fix of only one restriction per iteration (i.e., removing one machine from the eligible set) and the opposite, fixing the maximum number of possible restrictions over the selected shared job (i.e., leaving only one machine in the eligible set). The first approach is called *removing-one-machine* (ROM) and the second one is called *assign-one-machine* (AOM). In the following, each submodule is described in detail, introducing the specification and our implementations.

The distributed search The distributed search module, which is implemented by the worker agents, aims at producing local solutions s_w from the job assignation Y^* while extrapolating job information regarding the solutions produced. If an initial global solution s^I is received, the search process has to guarantee the same (or a higher) solution quality from the initial one ($F(\bigcup s_w) \leq F(s^I)$). Moreover, the module has to implement a rescheduling functionality when it is required to include a new set of shared-job J' to a local solution s_w and produce the rescheduled solution $s_{w,+J'}$. To produce a local solution s_w, each worker agent builds its $DataSet(w)$ from the Y^* and implements a search strategy that best fits the time constraints and the partial problem size. Then, each worker starts to analyze its produced s_w in order to identify the shared jobs that satisfy local job properties. Both steps are executed asynchronously by all of the worker agents $w \in A_w$. In our implementation, the solution s_w is calculated from the decomposition of s^I, while the rescheduling functionality is implemented as a local search procedure that inspects neighborhood solutions close to s_w that are generated from the insertion of a chosen shared-job j' over all the possible positions inside s_w. Without loss of generality, this implementation choice fixes the J' to be a singleton set j'. The local job property *Worst Local Job* (WLJ) proposed in (Nicolò et al. 2017) is used for the shared-job analysis. To formalize it, the following notation is introduced:

- $s_{w,-J'}$ is the local solution s_w after removing the set of assigned shared-jobs J' from their scheduled positions.
- $\Delta F^-_{s_w,-J'} = F(s_w) - F(s_{w,-J'})$ is the objective function variation once the jobs J' are removed from the local solution s_w.

From previous notation, WLJ of local solution s_w is defined as the total functions over the set of local solutions $\{\bigcup s_w\}$:

$$WLJ(s_w) = \underset{j' \in J^*}{\arg\max}\, \Delta F^-_{s_w,-j'}(s_w)$$

where J^* is the set of shared jobs that satisfy the current restrictions such that $\forall j' \in J^* : w \in M^*_{j'} \wedge |M^*_{j'}| > 1$.

The agreement The agreement module, which is implemented by worker agents, aims to choose a set of shared-job J' and define restricted sets over their eligible machine sets $M^*_{J'}$. To achieve this, the worker agents start the agreement process by sharing the information produced during the distributed search, namely, the local solutions with their job property analysis. After that, they agree on a set of chosen shared-jobs J', which are included in a specific s_w produced by worker w that satisfy global job properties that are related to all of the local solutions produced. After that, they agree on a set of chosen shared-jobs J' that are related to a specific s_w produced by worker w; these J' have to satisfy the global job properties that are related to all of the local solutions produced. For this set of shared-jobs J', a distributed rescheduling procedure is invoked, where J' is included in the $s_{w'}$ produced by other eligible workers ($w' \in M_{J'} \setminus w$). Thus, each worker starts a reschedule procedure to produce rescheduled local solutions $s_{w',+J'}$. If the whole set J' is not eligible for a worker w', just the subset composed of shared-jobs j' eligible for w' are rescheduled. Once the distributed reschedule is carried out, the workers involved share their rescheduled solutions $s_{w',+J'}$ so that a new agreement procedure starts. From s_w and $s_{w',+J'}$, an analysis of global job properties is performed in order to rank the eligible machines for J' with respect to the global properties. From this ranking, a subset of workers is chosen and removed from the set of eligible machines of J' through the definition of restrictions $M^*_{J'}$. Once the $M^*_{J'}$ is defined, the agreement terminates and the set of restrictions becomes *fixed*. This means that during the next iterations of the workflow, the machines removed from the eligible sets won't ever be analyzed for the set of agreed shared-jobs J'. For the evaluated implementations, the global job property *Worst Global Job* (WGJ) (Nicolò et al. 2017) is used. To formalize it, the following notation is used:

$$WGJ = \underset{j' \in J^*}{\arg\max} \max_{w \in A_w} \Delta F^-_{s_w,-j'}(s_w)$$

$$w_{WGJ} = \underset{w \in A_w}{\arg\max} \max_{j' \in J^*} \Delta F^-_{s_w,-j'}(s_w)$$

To describe the distributed rescheduling that is performed after the global job analysis, the following notations are introduced:

- $s_{w,+J'}$ is the rescheduled local solution s_w after job assignation of the set of shared-jobs J'
- $\Delta F^+_{s_w,+J'} = F(s_{w,+J'}) - F(s_w)$ is the the objective function variation over the local solution s_w with respect to its rescheduled local solution with shared-jobs J

After the distributed rescheduling, the analysis of global properties ranks the eligible worker for WGJ with respect to the global properties. Formally, a total order representing the ranking is defined as follows:

Definition 10. *The* ranking of eligible workers for WGJ *is a total order* $(M_{WGJ}, \overset{\Delta F}{\leq})$ *over the set of eligible machines* M_{WGJ}*, defined through the value* $\Delta F^-_{w_{WGJ},-WGJ}$ *for* w_{WGJ} *and* $\Delta F^+_{s_w,+WGJ}$ *for* $w \in \{M_{WGJ} \setminus w_{WGJ}\}$.

From the above defined total order $(M_{WGJ}, \overset{\Delta F}{\leq})$, the M^*_{WGJ} is the subset of M_{WGJ} that contains its first $\bar{m}$ workers, with $\bar{m} \in [\,1, \ldots, |M_{WGJ} - 1|\,]$. The value $\bar{m}$ depends on the implementations as follows:

$$\bar{m} = \begin{cases} |M_{WGJ} - 1| & \text{if ROM is selected} \\ 1 & \text{if AOM is selected} \end{cases}$$

Module overview As a general overview of the whole collaborative reasoner module, the pseudo-code in workflow 1 shows a workflow over the $\bar{m}$ value resulting in a concise way of reporting the two developed implementations of the framework: *removing-one-machine* (ROM) and *assign-one-machine* (AOM). The pseudo-code can be analyzed by grouped lines representing the introduced modules as follows: the loop of lines 1 to 3 represents the distributed search module for local solutions; the lines 4 to 6 represent the first invocation of the agreement module for local properties; the loop of lines 7 to 9 represents the distributed rescheduling; and the lines 10 to 14 represent the final agreement over the global job and set of restrictions.

4.4 The solution builder

The solution builder module is implemented by the supervisor agent in order to collect the local solutions s_w and agreed $\bigcup M^*_{J'}$ from the worker agents to build a final global

Workflow 1: Collaborative Reasoner

```
input  : $Y^*$ , $s^I$ , $\bigcup M_j^*$
output: $s_w$ , $M_{WGJ}^*$
forall $w \in A_w$ do
    $s_w \leftarrow w.Search(Dataset(\bigcup Y_{jw}^*),\ s^I,\ \bigcup M_j^*)$
    $WLJ(s_w) \leftarrow w.FindWLJ(s_w)$
forall $w \in A_w$ do
    $w.SharingOf(s_w \wedge WLJ, A_w)$
$WGJ \leftarrow findWGJ(\bigcup WLG(s_w))$
forall $w \in M_{WGJ} \setminus w_{WGJ}$ do
    $s_{w,+WGJ} \leftarrow w.Reschedule(s_w, WGJ)$
    $\Delta F^+_{s_w,+WGJ} \leftarrow w.Calculate\Delta(s_w, s_{w,+WGJ})$
forall $w \in M_{WGJ} \setminus w_{WGJ}$ do
    $w.SharingOf(s_{w,+WGJ} \wedge \Delta F^+_{s_w,+WGJ'}, M_{WGJ})$
$(M_{WGJ}, \overset{\Delta F}{\leq}) \leftarrow constructOrder(M_{WGJ}, \bigcup \Delta F)$
$M_{WGJ}^{ORD} \leftarrow M_{WGJ}.Sort(\overset{\Delta F}{\leq})$
$M_{WGJ}^* \leftarrow M_{WGJ}^{ORD}.KeepFirst(\bar{m})$
```

solution s^F, while restricted sets $\bigcup M_{j'}^*$ are propagated to the centralized solver module. For the proposed implementation, the building of global solutions is straightforward, while the restriction propagation for the centralized solver is implemented as the removal of elements from the eligible machine set that is embedded in the chromosome of the GA. In this way, the module can reuse the previously calculated individuals from the population of the GA.

5 Evaluation

In this section, an empirical evaluation of the Multi-Agent system is performed by analyzing the behavior of framework implementations in terms of efficiency, scalability, and solution quality. The evaluation is performed on a set of instances that is generated from a statistical study of real data acquired from a large plastic injection moulding factory of a leading company in the sector of supplying plastic trigger sprayers and pump dispensers (Paolucci, Anghinolfi, and Tonelli 2015). These instances are grouped by the number of jobs n and involved machines m for a total of 4 classes, each of which contains 125 problem instances. For the evaluation, all of the instances were executed on a 2.4 GHz Intel Core 2 Duo, limiting the computational time to 600 seconds, as imposed by the case study requirement. The weights expressing the relative importance of the objectives in (2) were fixed to $\Pi_1 = 0.6$, $\Pi_2 = 0.35$, and $\Pi_3 = 0.05$ according to the preference elicitation method introduced in (Paolucci, Anghinolfi, and Tonelli 2015). The average values obtained from the two framework implementations, ROM and AOM, were compared against two centralized approaches: the MILP model proposed in (Paolucci, Anghinolfi, and Tonelli 2015) (MILP) and the Genetic Algorithm (GA) proposed in (Nicolò et al. 2017), which is also used in the centralized solver module. Furthermore, our approaches were compared against the Multi-Agent System (MAS) proposed in (Tonelli et al. 2016). Section 5 shows

Instances		Multi-objective value				
		Centralized		Distributed		
n	m	MILP	GA	MAS	ROM	AOM
30	4	0,01997	0,03320	0,02663	0,03286	0,03284
50	6	0,01585	0,02811	0,02056	0,02752	0,02753
100	10	0,01305	0,02698	0,01780	0,02576	0,02559
250	20	0,05677	0,01428	0,03982	0,01197	0,01012

Table 1: Average multi-objective values reached in the different classes of instances in the 600-second timeout.

Objectives	GA	MAS	Framework	
			ROM	AOM
MO	-74,85%	-29,85%	-79,08%	-82,18%
TWT	-40,59%	-12,55%	-41,27%	-41,13%
EN	-2,92%	-4,95%	-2,47%	-2,42%
ST	60,11%	16,48%	48,6%	40,48%

Table 2: Percentage variation from MILP for the average objective functions values reached in the largest instance class.

the results with best values shaded in grey. It can be observed that the MILP model maintained the best behavior for small instances (30, 50, and 100 jobs), while the Multi-agent framework had better behavior for the largest instances (250 jobs). This is due to the fact that th eMILP model was able to obtain the optimal solution within the given timeout (600 seconds) in most of the small instances. However, as the problem dimension increased, both implementations of the Multi-Agent framework were able to achieve a better average optimized solution. It must be taken into account that the class of instances that is the closest to real problems is the largest one with $n = 250$ and $m = 20$. In fact, real instances in the injection air moulding industry scenario are composed of more that 500 jobs and 90 machines (Paolucci, Anghinolfi, and Tonelli 2015). Moreover, it can be observed that the values were close to 0, since (2) uses a minimum deviation method, where each of the three single objectives is compared with the best solution found by solving each one individually. Section 5 shows the percentage variation of each evaluated approach versus the MILP in the largest instance class of all the objectives (MO, TWT, EN, and ST). These results motivated the adoption of the proposed Multi-Agent framework for energy-aware scheduling due to its overall improvements while decreasing the energy consumption. It must be taken into account that each single objective is weighted by Π_i values according to the preferences given by experts. Therefore, the total setup time is almost ignored by the fitness function of the metaheuristics during the search process. As shown in Sections 5 and 5, ROM and AOM outperformed other approaches for large instances. Thus, to extrapolate more insights, a new class of randomly generate instances with a similar dimension of a real case ($n = 500$, $m = 40$) was generated and a comparative study of their asymptotic behavior was summarized and compared with the GA in Figure 2. The plot shows the progression of averaged objective values throughout the entire search process before a large timeout was reached (i.e., 3600 seconds is set as the asympototic value for time). It can be

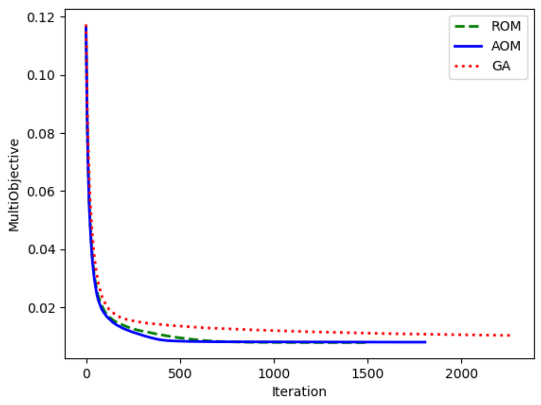

Figure 2: Asymptotic behavior comparison of GA versus ROA and AOM for the randomly generated instance ($n = 500$, $m = 40$) with 3600 seconds timeout.

		TWT	EN	ST	MO
Π^A	GA	10973646,98	37361,99	842,75	0,015084401
	ROM	10947597,94	37418,68	812,96	0,013630537
	AOM	10896557,66	37570,32	774,23	0,012468685
Π^B	GA	11114939,52	36927,08	848,94	0,021587188
	ROM	11039597,95	36932,21	822,34	0,02000914
	AOM	11056223,71	36985,63	795,12	0,018871841
Π^C	GA	13139426,46	42238,66	644,22	0,124417968
	ROM	12886306,94	42551,58	606,12	0,113732844
	AOM	12871967,53	44736,68	587,06	0,106763399
Π^D	GA	11031729,01	37099,19	840,09	0,018133667
	ROM	10987348,45	37174,17	818,46	0,017200711
	AOM	10988612,36	37225,67	789,80	0,016128346

Table 3: Average multi-objective values from four decision settings of the largest instance class in a 300-second timeout.

observed that each approach performed a different number of iterations within the given timeout. This is due to the fact that each approach has different complexity within each iteration. GA was able to carry out more iterations. However, it ended with worse solution quality with respect to AOM and ROM, showing the trade-off between the improvement from a collaborative reasoning versus a larger number of iterations. Moreover, by focusing on our implementations, AOM reached a better solution quality in a shorter amount of time; however, after a time threshold, ROM obtained better solutions. These results are not outlined in Section 5 due to the small timeout. Therefore, no specific framework implementation can be outlined as the best one but rather must be chosen with respect to the timeout value imposed by the non-functional constraints of the problem. For uniformity of comparison, the above evaluations used the decision weights to guide the whole search as defined in (Paolucci, Anghinolfi, and Tonelli 2015). Nevertheless, to extend the evaluation and to better assess the proposed implementation while keeping GA as the baseline, three new sets of weights were evaluated to analyze the behavior of our approaches in different scenarios: • The initial case: $\Pi^A = \{\Pi_1^A = 0.60, \Pi_2^A = 0.35, \Pi_3^A = 0.05\}$, • Giving more importance to energy: $\Pi^B = \{\Pi_1^B = 0.35, \Pi_2^B = 0.60, \Pi_3^B = 0.05\}$, • Giving the same importance to all objectives: $\Pi^C = \{\Pi_1^C = 0.34, \Pi_2^C = 0.33, \Pi_3^C = 0.33\}$, • Giving the same importance to tardiness and energy: $\Pi^D = \{\Pi_1^D = 0.475, \Pi_2^D = 0.475, \Pi_3^D = 0.05\}$. Section 5 summarizes the average results for the largest class of instances with a timeout of 300 seconds. Due to the tight timeout, AOM always outperformed the other evaluated approaches (AOM and GA) in the multi-objective based function. However, GA and ROM had better behavior than AOM in all scenarios in terms of energy consumption, while AOM had better performance in terms of setup time. Thus, a mixture of the proposed approaches can provide a promising technique that could be developed further.

6 Conclusions and Future Works

Several multi-agent approaches have been developed to manage scheduling problems. In this paper, a multi-agent framework has beeen proposed to solve energy-aware scheduling problems in order to be applied to a real case of the injection air moulding industry with intense energy consumption. Thus, a set of collaborative agents work in a distributed environment to reach an agreement in order to minimize several objective functions: total job tardiness, total setup time, and energy consumption. From framework generality, two specific implementations were developed and tested against centralized and distributed state-of-the-art approaches on a real case dataset of the problem. The results show the efficiency of framework implementations as a valuable choice for tackling sustainable scheduling problems. Also, meaningful insight has been gained regarding the trade-off between time spent on the proposed collaborative reasoning versus increasing the number of iterations, which is the approach that is commoly used to improve performance in many metaheuristics. An analysis of the design choices made by the original authors for the data set under evaluation has shown some limitations: the multi-objective criteria to express the solution quality is defined as a scalarized weighted function; and the time values of a problem instance are expressed as sensitive decimal numbers, which makes a constraint programming evaluation through a commercial solver unfeasible. These limitations suggest possible lines for future works: the design of a benchmark for the problem to enable CP evaluations; the use of different multi-objective criteria (e.g., pareto front, ϵ-constraint, etc.); the exploration of new implementations for the proposed framework (i.e., different metaheuristics for the centralized solver module, new job properties to guide agreement, etc.). Another interesting possibility is to consider different energy-aware problems: problems that allow energy consumption variation in relation to the processing mode (Zhang and Chiong 2016); scenarios that consider shifting the energy costs (Grimes et al. 2014); or exploiting the reactivity of the multi-agent system for dynamic energy-aware scheduling (Zhai et al. 2017).

Acknowledgements

The paper has been partially supported by the Spanish research project TIN2016-80856-R and TIN2015-65515-C4-1-R, and the ACIF/2017 Program of the Conselleria d'Educació (Generalitat Valenciana).

References

Agnetis, A.; Billaut, J.-C.; Gawiejnowicz, S.; Pacciarelli, D.; and Soukhal, A. 2014. *Multiagent Scheduling: Models and Algorithms*. Springer Science & Business Media.

Allahverdi, A. 2015. The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research* 246(2):345–378.

Behnamian, J. 2016. Graph colouring-based algorithm to parallel jobs scheduling on parallel factories. *International Journal of Computer Integrated Manufacturing* 29(6).

Bruzzone, A.; Anghinolfi, D.; Paolucci, M.; and Tonelli, F. 2012. Energy-aware scheduling for improving manufacturing process sustainability: a mathematical model for flexible flow shops. *CIRP Annals-Manufacturing Technology* 61(1):459–462.

Che, A.; Zhang, S.; and Wu, X. 2017. Energy-conscious unrelated parallel machine scheduling under time-of-use electricity tariffs. *Journal of Cleaner Production* 156(Supplement C):688 – 697.

Dai, M.; Tang, D.; Giret, A.; Salido, M. A.; and Li, W. D. 2013. Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. *Robotics and Computer-Integrated Manufacturing* 29(5):418–429.

Gong, X.; der Wee, M. V.; Pessemier, T. D.; Verbrugge, S.; Colle, D.; Martens, L.; and Joseph, W. 2017. Energy- and labor-aware production scheduling for sustainable manufacturing: A case study on plastic bottle manufacturing. *Procedia CIRP* 61(Supplement C):387 – 392. The 24th CIRP Conference on Life Cycle Engineering.

Graham, R. L.; Lawler, E. L.; Lenstra, J. K.; and Kan, A. R. 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics* 5:287–326.

Grimes, D.; Ifrim, G.; O'Sullivan, B.; and Simonis, H. 2014. Analyzing the impact of electricity price forecasting on energy cost-aware scheduling. *Sustainable Computing: Informatics and Systems* 4(4):276–291.

Han, D.; Yang, B.; Li, J.; Wang, J.; Sun, M.; and Zhou, Q. 2017. A multi-agent-based system for two-stage scheduling problem of offshore project. *Advances in Mechanical Engineering* 9(10):1687814017720882.

Lawler, E. L. 1977. A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness. *Annals of discrete Mathematics* 1:331–342.

Liu, N.; Abdelrahman, M. A.; and Ramaswamy, S. 2007. A complete multiagent framework for robust and adaptable dynamic job shop scheduling. *IEEE Transactions on Systems Man, and Cybernetics-Part C: Applications and Reviews* 37(5):904–916.

Liu, G.-S.; Yang, H.-D.; and Cheng, M.-B. 2017. A three-stage decomposition approach for energy-aware scheduling with processing-time-dependent product quality. *International Journal of Production Research* 55(11):3073–3091.

Lu, N.-y.; Gong, G.-x.; Yang, Y.; and Lu, J.-h. 2012. Multi-objective process parameter optimization for energy saving in injection molding process. *Journal of Zhejiang University SCIENCE A* 13(5):382–394.

Manupati, V. K.; Rajyalakshmi, G.; Chan, F. T. S.; and Thakkar, J. J. 2017. A hybrid multi-objective evolutionary algorithm approach for handling sequence- and machine-dependent set-up times in unrelated parallel machine scheduling problem. *Sādhanā* 42(3):391–403.

Martin, S.; Ouelhadj, D.; Beullens, P.; Ozcan, E.; Juan, A. A.; and Burke, E. K. 2016. A multi-agent based cooperative approach to scheduling and routing. *European Journal of Operational Research* 254(1):169–178.

Newman, S.; Nassehi, A.; Imani-Asrai, R.; and Dhokia, V. 2012. Energy efficient process planning for cnc machining. *CIRP Journal of Manufacturing Science and Technology* 5(2):127 – 136.

Nicolò, G.; Salido, M. A.; Ferrer, S.; Giret, A.; and Barber, F. 2017. A multi-agent approach using dynamic constraints to solve energy-aware unrelated parallel machine scheduling problem with energy-dependent and sequence-dependent setup time. *COPLAS 2017* 31.

Paolucci, M.; Anghinolfi, D.; and Tonelli, F. 2015. Facing energy-aware scheduling: a multi-objective extension of a scheduling support system for improving energy efficiency in a moulding industry. *Soft Computing* 1–12.

Plitsos, S.; Repoussis, P. P.; Mourtos, I.; and Tarantilis, C. D. 2017. Energy-aware decision support for production scheduling. *Decision Support Systems* 93(Supplement C):88 – 97.

Seow, Y., and Rahimifard, S. 2011. A framework for modelling energy consumption within manufacturing systems. *CIRP Journal of Manufacturing Science and Technology* 4(3):258–264.

Tonelli, F.; Bruzzone, A.; Paolucci, M.; Carpanzano, E.; Nicolò, G.; Giret, A.; Salido, M.; and Trentesaux, D. 2016. Assessment of mathematical programming and agent-based modelling for off-line scheduling: Application to energy aware manufacturing. *CIRP Annals-Manufacturing Technology*.

Tonelli, F.; Evans, S.; and Taticchi, P. 2013. Industrial sustainability: challenges, perspectives, actions. *International Journal of Business Innovation and Research* 7(2):143–163.

Zhai, Y.; Biel, K.; Zhao, F.; and Sutherland, J. W. 2017. Dynamic scheduling of a flow shop with on-site wind generation for energy cost reduction under real time electricity pricing. *CIRP Annals* 66(1):41 – 44.

Zhang, R., and Chiong, R. 2016. Solving the energy-efficient job shop scheduling problem: a multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption. *Journal of Cleaner Production* 112:3361–3375.