Energy-aware Multiple State Machine Scheduling for Multiobjective Optimization*

Angelo Oddi[1], Riccardo Rasconi[1], and Miguel A. González[2]

[1] Institute of Cognitive Sciences and Technologies, ISTC-CNR, Italy
{angelo.oddi,riccardo.rasconi}@istc.cnr.it
[2] Department of Computing, University of Oviedo, Spain
mig@uniovi.es

Abstract. Optimising the energy consumption is one of the most important issues in scheduling nowadays. In this work we consider a multi-objective optimisation for the well-known job-shop scheduling problem. In particular, we minimise the makespan and the energy consumption at the same time. We consider a realistic energy model where each machine can be in *Off*, *Stand-by*, *Idle* or *Working* state. We design an effective constraint-programming approach to optimise both the energy consumption and the makespan of the solutions. Experimental results illustrate the potential of the proposed method, outperforming the results of the current state of the art in this problem.

Keywords: Constraint-programming · job-shop scheduling · energy considerations · multi-objective optimisation.

1 Introduction

The job shop is a scheduling problem widely studied in the literature due to the fact that it is a model which is close to many real production environments. It is proven that the job shop is NP-hard, and so its resolution is very complex. In the literature we can find many different solving approaches for the job shop, from exact methods to all kinds of meta-heuristic algorithms.

Although the makespan is the most studied objective function, energy considerations are increasingly important nowadays, mainly for economical and environmental reasons. In fact, we can find a number of papers addressing the energy-efficient job shop. For example, Zhang and Chiong [12] try to minimise both the weighted tardiness and the energy consumption in a job shop where the processing mode of operations can be modified. Another approach is that of Liu et al. [7], where it is considered a simple energy model where the machines can only be in *Working* or in *Idle* state. González et al. [4] improve the

* This research has been supported by the Spanish Government under research project TIN2016-79190-R. ISTC-CNR authors were supported by the ESA Contract No. 4000112300/14/D/MRP "Mars Express Data Planning Tool MEXAR2 Maintenance".

results reported in [7] by using a hybrid evolutionary meta-heuristic and also a constraint-programming approach. One problem with the last two papers is that the considered energy model is not too realistic. The model proposed by May et al. [8] is much more interesting, as the machines can be either in the *Idle*, *Working*, *Off*, or switched to a *Stand-by* state.

In this paper we consider this last energy model and try to minimize at the same time the makespan and the energy consumption in a job shop. Although some multi-objective works consider weighted or lexicographical approaches, probably the most interesting approaches are those based on the Pareto Front.

In particular, we have designed a constraint-based procedure to minimise both the makespan and the energy consumption, within a well-studied multi-objective optimisation method to generate the whole Pareto (i.e., the ϵ-constraint method [9]). The contribution of the paper is the following. We propose a constraint-based model where: (i) we add as decision variables the states of the machines during the no-working periods (i.e., *Idle*, *Off*, or *Stand-by* states); (ii) we introduce energy aware constraints that exploit the total order of activities on each machine, as well as the lower bound on each machine's total execution time, ultimately implementing a new propagation rule. We show that this new model exhibits interesting performances, outperforming both the results obtained in [8], and the more recent results obtained in [10].

This paper is organised as follows: Section 2 formulates the problem at hand and Section 3 describes the solving methods. Then, in Section 4 we analyse our proposals and we compare them with the state-of-the-art algorithms [8, 10], and finally in Section 5 we report some conclusions and ideas for future work.

2 Problem formulation

The job shop scheduling problem (JSP) consists on scheduling a set of N jobs, $J = \{J_1, \ldots, J_N\}$ in a set of M machines or resources, $R = \{R_1, \ldots, R_M\}$. Each of the jobs J_i consists of n_i tasks $(\theta_{i1}, \ldots, \theta_{in_i})$ that must be scheduled exactly in that particular order. Each task requires a given resource during all its processing time. Additionally, no preemption is allowed, so when a resource starts processing a task, it cannot be interrupted until it ends. Moreover, resources can at most process one task at a time. The objective of the problem is to minimise some objective functions subject to the described precedence and capacity constraints. Although we have denoted the tasks as θ_{ij} in this problem definition, in the following we will denote them by a single letter, if possible, in order to simplify the expressions. We denote by Ω the set of tasks, by p_u the processing time of task u, by r_u the resource required by task u, and by s_u the starting time of task u (which needs to be determined).

As we have seen, the JSP has precedence constraints, defined by the routing of the tasks within the jobs, that translate into linear inequalities: $s_u + p_u \leq s_v$, where v is the next task to u in the job sequence. The problem has also capacity constraints, as the resources can only process one task at a time, and they translate into disjunctive constraints: $(s_u + p_u \leq s_v) \vee (s_v + p_v \leq s_u)$, where u

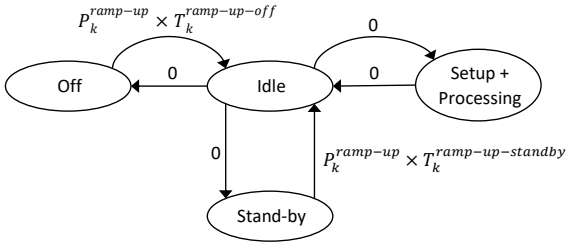


Fig. 1. State diagram for a machine, indicating the energy consumed in each transition

and v are tasks requiring the same resource. The objective is to build a feasible schedule, i.e. determine a starting time for each task such that all constraints are fulfilled. In the following, given a feasible schedule, we will denote with PJ_v and SJ_v the predecessor and successor of v, respectively, in the job sequence, and with PM_v and SM_v the predecessor and successor of v, respectively, in its resource sequence. In addition, we will denote with α_k and ω_k the first and last operations respectively on machine R_k in the considered schedule.

The goal of the present analysis is the minimisation of both the energy consumption and the overall completion time, or *makespan*. In general, for a minimization problem with two objective functions f_i $(i = 1, 2)$, a solution S is said to be *dominated* by another solution S', denoted $S' \prec S$, if and only if for each objective function f_i, $f_i(S') \leq f_i(S)$ and there exists at least one i such that $f_i(S') < f_i(S)$. However, the possibly conflicting nature of these two objectives may prevent the existence of a unique solution S^* that is optimal w.r.t. both the objectives. Therefore, in this work we are interested in the set of all optimal "tradeoffs", which are known as the *Pareto optimal* solutions, i.e. solutions such that the improvement of one objective necessarily implies the worsening of the other objective. The Pareto front PS^* is the set of solutions S, such that for each $S \in PS^*$ there is no solution S' which dominates S $(S' \prec S)$.

The makespan is the first objective function and corresponds to the maximum completion time of the schedule, that is $\max_{u \in \Omega}\{s_u + p_u\}$. About the second objective the energy model is taken from [8], where it is supposed that a resource can be in five different states: *Off*, *Stand-by*, *Idle*, *Setup* or *Working*. However, May et al. in their experiments from [8] consider together the times and energy consumption of the *Working* and *Setup* states; as a consequence, we can consider a total of four possible states (see Figure 1). The power consumption in each state for a given resource R_k is denoted by P_k^{idle}, $P_k^{stand\text{-}by}$ and $P_k^{working}$, whereas if the machine is *Off* it consumes no power. Additionally, we assume that the machine can instantly switch from *Idle* to *Stand-by*, *Off* or *Working*, consuming no power. On the other hand, switching from *Off* to *Idle* requires an amount of $T_k^{ramp\text{-}up\text{-}off}$ time units, whereas switching from *Stand-by* to *Idle* requires $T_k^{ramp\text{-}up\text{-}stand\text{-}by}$ time units. In both cases, the power consumed when ramping up is denoted by $P_k^{ramp\text{-}up}$. In Figure 1 we show the considered state diagram, which is the same for each machine. Also, we assume that all machines do not consume any energy before the processing of its first task assigned. It is easy to see that in the

job shop scheduling problem, each resource must always process the same set of tasks, and so the working energy consumption is the same in every possible schedule. Therefore, following [8], in order to reduce the energy consumption we consider the WEC (Worthless Energy Consumption) measure as the second objective function to minimize, which is defined as follows:

$$WEC = \sum_{k=1,\ldots,M} (P_k^{idle}\, t_k^{idle} + P_k^{stand\text{-}by}\, t_k^{stand\text{-}by}) + \sum_{k=1,\ldots,M} P_k^{ramp\text{-}up}\, (n_k^{standby}\, T_k^{ramp\text{-}up\text{-}standby} + n_k^{off}\, T_k^{ramp\text{-}up\text{-}off}) \quad (1)$$

where t_k^{idle} is the total amount of time spent by R_k in *Idle* state, $t_k^{stand\text{-}by}$ is the total amount of time spent by R_k in *Stand-by* state, $n_k^{standby}$ is the number of times that resource R_k transitions from *Stand-by* to *Idle* state, and finally n_k^{off} is the number of transitions from *Off* to *Idle*.

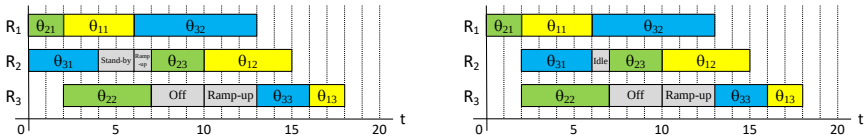
To the aim of assessing how the power consumption of the machines may vary depending on the different states to which they are allowed to transition, we follow the analysis performed in [8], taking into account two different machine behavior *policies*, which we will respectively call $P3$ and $P4$ as described in the following. The $P3$ policy is implemented by switching the machines on at their first operation and switching them off at their last, with the possibility to switch them on and off from the *Idle* state, between any pair of consecutive tasks belonging to the production batch (see Figure 1). The $P4$ policy is similar to the previous one, with the addition of the *Stand-by* state. According to the $P4$ policy, each machine can transition from the *Idle* state to the *Stand-by* state during the production batch, whenever such transition is energetically convenient over both switching the machine on and off again, and leaving it in the *Idle* state. In [8] two more policies called $P1$ and $P2$ are investigated, but such policies are not taken into account in this work because they are very simple and hence not of great interest for our purposes.

According to K. Baker [2], the makespan is a regular performance measure, which means that it can be increased only by increasing at least one of the completion times in a given schedule. To optimize regular measures it is enough to consider "left-shift schedules", i.e. schedules that are built from a given ordering of the tasks, in such a way that each operation starts in the earliest possible time after all the preceding tasks in the ordering. As opposed to the makespan, the WEC is a non-regular measure, and it can sometimes be decreased by increasing the completion time of some tasks while leaving the other tasks unmodified.

To better illustrate the problem we present a small toy example. Consider an instance with 3 jobs (with 3 tasks for each job) and 3 resources. The processing times are the following: $p_{\theta_{11}} = 4$, $p_{\theta_{12}} = 5$, $p_{\theta_{13}} = 2$, $p_{\theta_{21}} = 2$, $p_{\theta_{22}} = 5$, $p_{\theta_{23}} = 3$, $p_{\theta_{31}} = 4$, $p_{\theta_{32}} = 7$, $p_{\theta_{33}} = 3$. The required resources are as follows: $r_{\theta_{11}} = R_1$, $r_{\theta_{12}} = R_2$, $r_{\theta_{13}} = R_3$, $r_{\theta_{21}} = R_1$, $r_{\theta_{22}} = R_3$, $r_{\theta_{23}} = R_2$, $r_{\theta_{31}} = R_2$, $r_{\theta_{32}} = R_1$, $r_{\theta_{33}} = R_3$. Also, consider the following values for every machine $k \in \{1, 2, 3\}$: $P_k^{working} = 10kW$, $P_k^{idle} = 6kW$, $P_k^{stand\text{-}by} = 4kW$, $P_k^{ramp\text{-}up} = 8kW$, $T_k^{ramp\text{-}up\text{-}off} = 3$ and $T_k^{ramp\text{-}up\text{-}stand\text{-}by} = 1$.

Figure 2(a) shows a feasible solution for this instance. In fact it is a "left-shift schedule" (see Section 2). This schedule has a makespan of 18 and a WEC of 40 (16 from R_2 plus 24 from R_3). In resource R_2 we have decided to switch the machine to *Stand-by* state between the end of θ_{31} and the beginning of θ_{23}, because in this case it adds 16 units to the WEC, whereas if switched *Off* it would add 24 units and if it remained *Idle* it would add 18 units. Using the same reasoning we switch R_3 off between the end of θ_{22} and the beginning of θ_{33}.

This "left-shift schedule" can be improved by delaying some tasks. As an example, Figure 2(b) shows the same solution after delaying task θ_{31}. Now there is only one time unit between the end of θ_{31} and the beginning of θ_{23}, and so the best option is to leave the machine in *Idle* state. The makespan is still 18 but the WEC is reduced from 40 to 30.



(a) A feasible solution, WEC = 40. (b) Improved solution, WEC = 30.

Fig. 2. Improving a solution by delaying one task.

3 The proposed solving method

As we have seen in the previous section, the WEC is a non-regular performance measure. Moreover, the work [8] only considers "left-shift schedules", while we have seen that they can be improved by delaying some tasks, in order to reduce the total energy consumption. In this section we describe a procedure that takes into account the non-regularity of the WEC objective such that an approximation of the Pareto front is generated by a Constraint Programming (CP) procedure. It is worth noting that the proposed CP approach is in principle able to find an optimal WEC value if given sufficient computational time (we do not provide any formal proof about this property).

3.1 Energy optimisation: a Constraint Programming approach

Constraint Programming (CP) is a declarative programming paradigm [1]. A constraint program is defined as a set of *decision variables*, each ranging on a discrete domain of values, and a set of *constraints* that limit the possible combination of variable-value assignments. After a *model* of the problem is created, the solver interleaves two main steps: *constraint propagation*, where inconsistent values are removed from variable domains, and *search*. CP is particularly suited

for solving scheduling problems where the decision variables are associated to the problem operations. In particular, each operation variable a is characterised at least by two features: s_a representing its start time, and p_a representing its duration. For scheduling problems, a number of different *global constraints* have been developed, the most important being the `unary-resource` constraint [11] for modelling simple machines, the `cumulative` resource constraint [6] for modelling cumulative resources (e.g., a pool of workers), and the `reservoir` [5] for modelling consumable resources (e.g., a fuel tank). In particular, given `unary-resource(`A`)`, the constraint holds if and only if all the operations in the set A never overlap at any time point. A number of propagation algorithms are embedded in the `unary-resource` constraint for removing provably inconsistent assignments of operation start-time variables.

We describe a Constraint Programming (CP) model based on the problem defined in Section 2, where the main *decision variables* are the start times s_a of the operations $a \in \Omega$ characterized by a processing time p_a. Each start time s_a ranges in the interval $[0, H - p_a]$, where H is the problem's horizon. The set of decision variables is then extended with the start times s_{OnOff_k} of the $OnOff_k$ intervals, where each $OnOff_k$ interval is defined as spanning over all the operations executed on machine k. Hence, the s_{OnOff_k} variable represents the first instant when machine k is turned on. The model, whose utilisation will be described in the experimental section (Section 4), is built on top of the IBM-ILOG CPLEX Optimization Studio CP Optimizer and its details are as follows.

Let O_k be the set of problem operations assigned to machine $k = 1, \ldots, M$ and U_k be a set of auxiliary *unit-duration operations*, assigned to a dummy unary machine mirroring k (it is worth noting that the two sets O_k and U_k represent separate processing orders of activities). The introduction of the auxiliary set of operations U_k [3] is necessary to represent the position of each activity $a \in O_k$ in the processing orders imposed among the operations assigned to each machine $k \in R$. More concretely, the auxiliary unit-duration operations indirectly implement the definition of a *successor function* SM_a (returning the successor of each operation a for each total order imposed on the set of operations O_k assigned to a machine k). To the best of our knowledge, this workaround is necessary because we want to use the native OPL construct to implement the global constraints `unary-resource(`O_k`)` for efficiency reasons, and the successor function is not natively present in the OPL language (see IBM ILOG CPLEX Optimization Studio OPL Language Reference Manual, Version 12 Release 7.1).

Operationally, the set of *unit-duration* operations $u \in U_k$ can be assigned to the dummy machine k (in the same fashion of the operations a) so that, for each processing order imposed on a machine k, $a_0 \prec a_1 \prec \ldots \prec a_i \prec \ldots \prec a_M$, an identical order is imposed on the unit-duration operations $u_0 \prec u_1 \prec \ldots \prec u_i \prec \ldots \prec u_M$. In this manner, the position i of the operation a_i coincides with the start-time value of the unit-duration operation u_i. For the reasons above,

[3] We were inspired to adopt this solution by a post on a discussion board on the website www.or-exchange.com about the explicit representation of an interval position in a OPL `sequence`. This discussion board does not seem available anymore.

the starting times s_u of the operations $u \in U_k$ must be added to the model as additional set of *decision variables*. In addition, a specific global constraint is added in the CP model given below to impose the same order among the activities in the sets O_k and U_k, see constraints (3i).

The definition (2a) represents the successor function SM_p, such that the position of the operation $p \in O_k$ coincides with the start-time value $s_{u^{(p)}}$ of its corresponding unit-duration operation $u^{(p)} \in U_k$, and the successor q (if exists) corresponds to the unary activity $u^{(q)} \in U_k$, such that $s_{u^{(q)}} = s_{u^{(p)}} + 1$. Whereas, according to Section 2, the energy objective *WEC* (2c) is the sum of the unload energy consumption E^k_{pq} (i.e., when a machine is *Idle*, switched *Off*, or switched to a *Stand-by* state) of each pair of contiguous operations (p, q) assigned on the same machine k (2b), where $d_{pq} = s_q - e_p$ is the difference between q's start time and p's end time. The makespan objective C_{max} is described at line (2d).

$$SM_p = \begin{cases} q & \exists u^{(q)} \in U_k : s_{u^{(q)}} = s_{u^{(p)}} + 1 \\ nil & \text{otherwise} \end{cases} \tag{2a}$$

$$\begin{aligned} E^k_{pq} = min\{&P^{idle}_k \, d_{pq}, \\ &P^{stand\text{-}by}_k \, (d_{pq} - T^{ramp\text{-}up\text{-}standby}_k) + P^{ramp\text{-}up}_k \, T^{ramp\text{-}up\text{-}standby}_k, \\ &P^{ramp\text{-}up}_k \, T^{ramp\text{-}up\text{-}off}_k\} \end{aligned} \tag{2b}$$

$$WEC = \sum_{k=1,\ldots,M} \sum_{\substack{p \in O_k, \\ q = SM_p,\, q \neq nil}} E^k_{pq} \tag{2c}$$

$$C_{max} = \max_{a \in \Omega} \{s_a + p_a\} \tag{2d}$$

Once all the necessary definitions have been provided and all the variables have been introduced, we present the CP model (optimisation criteria and constraints). Line (3a) represents the lexicographic minimisation of the objective pair *WEC* and C_{max} with the energy *WEC* as primary objective. According to the implemented ϵ-constraint method [9] for calculating the Pareto set, we optimise the energy *WEC*, while we impose an upper bound to the other objective C_{max} in the form $C_{max} \leq C_\epsilon$ (see (3b)). The constraints in (3c) represent the linear orderings imposed on the set of operations Ω by the set of jobs J. Constraints (3d) impose to the set O_k of operations requiring machine k to be contained in the *spanning* operations $OnOff_k$, $k = 1, \ldots, M$. More specifically, for each operation $v \in O_k$, the following constraints $s_{OnOff_k} \leq s_v$ and $s_v + p_v \leq s_{OnOff_k} + p_{OnOff_k}$ hold, such that operation $OnOff_k$ starts together with the *first* present operation in O_k according to the order imposed on the k-th machine, and ends together with the *last* present operation.

Constraints (3f), (3g), and (3h) impose that the minimal energy is consumed between the end of the first and the beginning of the second task, for each pair of contiguous activities (p, q) on a resource k. These constraints rely on the assumption that $P^{stand\text{-}by}_k \leq P^{idle}_k \leq P^{ramp\text{-}up}_k$ and $T^{ramp\text{-}up\text{-}standby}_k \leq T^{ramp\text{-}up\text{-}off}_k$; under such assumptions, there are two cutoff values, $T^{idle\text{-}standby}_k$ and $T^{standby\text{-}off}_k$ (depicted in Figure 3), such that if $s_v - e_u \in [0, T^{idle\text{-}standby}_k]$ the minimal energy

state is *Idle*, when $s_v - e_u \in (T_k^{idle\text{-}standby}, T_k^{standby\text{-}off}]$ the minimal energy state is *Stand-by*, otherwise the minimal energy state is *Off*.

$$\text{lex min}\,(WEC, C_{max}) \tag{3a}$$

s.t. :

$$C_{max} \leq C_\epsilon \tag{3b}$$

$$s_v + p_v \leq s_{SJ_v} \quad v \in \Omega \setminus \{\theta_{1n_1}, .., \theta_{Nn_N}\} \tag{3c}$$

$$\texttt{span}(OnOff_k, O_k) \quad k = 1, .., M \tag{3d}$$

$$ed_p \in \{0, 1, 2\}\ p \in \Omega \tag{3e}$$

$$SM_p = q \wedge (ed_p = 0) \Rightarrow s_q - e_p \leq T_k^{idle\text{-}standby} \tag{3f}$$

$$SM_p = q \wedge (ed_p = 1) \Rightarrow s_q - e_p > T_k^{idle\text{-}standby} \wedge s_q - e_p \leq T_k^{standby\text{-}off} \tag{3g}$$

$$SM_p = q \wedge (ed_p = 2) \Rightarrow s_q - e_p > T_k^{standby\text{-}off} \tag{3h}$$

$$\texttt{same-sequence}(O_k, U_k)\ k = 1, \ldots, M \tag{3i}$$

$$s_u \leq (|O_k| - 1)\ u \in U_k;\ k = 1, \ldots, M \tag{3j}$$

$$\texttt{unary-resource}(O_k)\ k = 1, \ldots, M \tag{3k}$$

$$\texttt{unary-resource}(U_k)\ k = 1, \ldots, M \tag{3l}$$

$$\Delta t_k^{proc} + \Delta t_k^{standby} + \Delta t_k^{off} \leq C_{max}\ \ k = 1, \ldots, M \tag{3m}$$

We introduce a set of decision variables $ed_p \in \{0, 1, 2\}$, $p \in \Omega$ (constraint (3e)) representing the unload state (i.e., 0 when machine is *Idle*, 1 when it is switched to a *Stand-by* state, and 2 when switched *Off*) imposed on every pair of contiguous activities (p, q) on the same machine. The constraints in (3i) impose the same order between the activities in the two sets O_k and U_k by means of the global constraints $\texttt{same-sequence}(O_k, U_k)$. The constraints in (3j) bound the start-time value of each unit-duration operation u to $|O_k| - 1$ operations assigned to the machine k. (3k) and (3l) represents the non-overlapping constraints imposed by the machines M to the operations in O_k and U_k, through the global constraints $\texttt{unary-resource}(O_k)$ and $\texttt{unary-resource}(U_k)$, respectively.

Finally, (3m) represents the so-called *energy aware constraints* imposed on the subset of (energy) decision variables ed_p associated to each subset of operations O_k, $k = 1, \ldots, M$. The rational behind this constraints is the following: for each machine k, the set of operations O_k requiring that machine must be totally ordered. In addition, according to the values of the decision variables ed_u, with $u \in O_k$, a minimum (non zero) delay equal to $T_k^{ramp\text{-}up\text{-}standby}$ (when $ed_u = 1$) or $T_k^{ramp\text{-}up\text{-}off}$ (when $ed_u = 2$), must be inserted between the operation u and its successor (if it exists). Hence, each machine's total order has a lower-bound of the total execution time (from the start-time of the first operation to the end-time of the last one) which can be calculated as the sum of the three terms $\Delta t_k^{proc} + \Delta t_k^{stand\text{-}by} + \Delta t_k^{off}$, such that: $\Delta t_k^{proc} = \sum_{u \in O_k} p_u$ is the sum of the operation processing times in machine k; $\Delta t_k^{standby} = \sum_{u \in O_k,\, ed_u = 1} T_k^{ramp\text{-}up\text{-}standby}$ is the minimum total delay due to *Stand-by* states; $\Delta t_k^{off} = \sum_{u \in O_k,\, ed_u = 2} T_k^{ramp\text{-}up\text{-}off}$ is the minimum total delay due to *Off* states. Such lower-bound cannot be greater

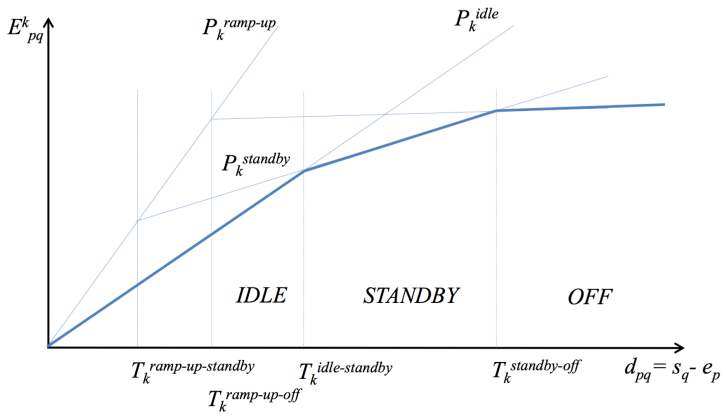


Fig. 3. Minimal energy consumption E^k_{pq} between two consecutive operations (p, q).

than the solution makespan C_{max}, hence decisions on the variables ed_u can be pruned according to the constraints (3m). This new propagation rule is the main innovation with respect to work [10], and we evaluate it in Section 4.

3.2 The bi-criterion ϵ-constraint method

A well-known multi-objective optimization method to generate the Pareto front is the ϵ-constraint method [9]. It works by choosing one objective function as the only objective and properly constraining the remaining objective functions during the optimisation process. Through a systematic variation of the constraint bounds, different elements of the Pareto front can be obtained.

Algorithm 1 presents the ϵ-constraint method for the case of a bi-criterion objective function $\mathbf{f} = (f^{(1)}, f^{(2)})$. The algorithm is used in the experimental section of the work and takes the following inputs: (i) the objective $\mathbf{f}$, (ii) the bounds $f^{(2)}_{min}$ and $f^{(2)}_{max}$ on the second component of the objective, and (iii) the decrement value δ. As previously mentioned, the method iteratively leverages a procedure provided in input to solve constrained optimization problems, i.e., the CP() procedure corresponding to the constraint programming model previously described. Note that we consider a slightly different ϵ-constraint method, such that the given CP procedure considers a lexicographic minimisation instead of single-objective minimisation problem, with $f^{(1)}$ as primary and $f^{(2)}$ as secondary key. The algorithm proceeds as follows: after initializing the constraint bound ϵ to the $f^{(2)}_{max}$ value, a new solution S is computed by calling CP() at each step of the *while* solving cycle. If S is not dominated by any of the existing solutions in the current Pareto front approximation P, then S is inserted in P, and all the solutions possibly dominated by S are removed from P. The

Algorithm 1 Bi-criterion ϵ-constraint method

Require: The objective $\mathbf{f}$, the bounds $f^{(2)}_{min}$ and $f^{(2)}_{max}$, and the decrement value δ
  $P \leftarrow \emptyset$;
  $\epsilon \leftarrow f^{(2)}_{max}$;
  while $\epsilon \geq f^{(2)}_{min}$ **do**
    $S \leftarrow \mathrm{CP}(\mathbf{f}, \epsilon)$;
    if $(S \neq nil) \wedge (\nexists S^{'} \in P : S^{'} \prec S)$ **then**
      $P \leftarrow (P \cup \{S\}) \setminus \{S^{'} \in P : S \prec S^{'}\}$;
    end if
    $\epsilon \leftarrow \epsilon - \delta$;
  end while
  return P

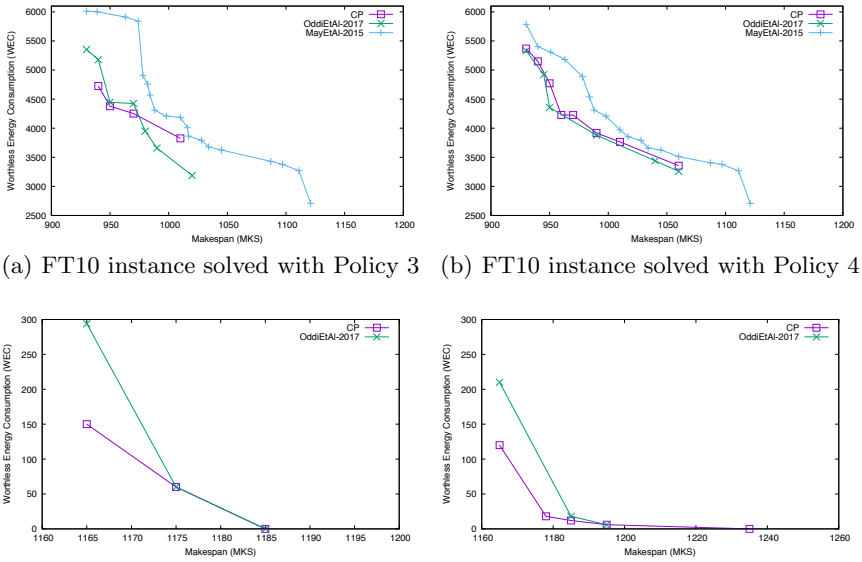
rationale behind this method is to iteratively tighten the constraint bound by a pre-defined constant δ at each step of the solving cycle.

4 Experimental results

In this section we will analyze the results we have obtained with our CP procedure, and compare such results with the state of the art in [8, 10]. In our work, we test our model against three well-known JSP instances called, respectively, FT06, FT10 and FT20 (as considered in [8]). These instances were introduced by Fisher and Thompson [3], and are characterized by different dimensions (*number-of-jobs* $\times$ *number-of-machines*): FT06 (6×6), FT10 (10×10), and FT20 (20×5). According to the literature, the optimal makespan of these instances is 55, 930 and 1165, respectively. The energy values for the machines are those described in the toy example of Section 2. In our tests, we have compared our results with those present in two recent works [8, 10] and related to the machine behavior policies $P3$ and $P4$ introduced in Section 2, as these are the most interesting from the energy minimization standpoint. From the analysis performed in Section 2, it is certain that the solutions obtained with the $P4$ policy will exhibit energy consumptions lower than or equal to those obtained with the $P3$ policy, due to the additional possibility of switching machines to *Stand-by* state.

Figure 4 graphically presents a comparison of the obtained results in instances FT10 and FT20 (results for FT06 instance can only be found in Tables 1 and 2) using policies P3 and P4. In particular, the plots labelled "*MayEtAl-2015*" and "*OddiEtAl-2017*" describe the Pareto front approximations reported in [8] and [10], respectively, whereas the plots labelled "*CP*" describe the Pareto front approximations obtained with our new constraint programming model using the set of *energy-aware constraints* described in Section 3.1.

In these tests, for the CP model we allowed for a maximum 5 minutes for each $FT06$ solution and a maximum 15 minutes for each $FT10$ and $FT20$ solutions. The proposed CP model has been implemented on the IBM-ILOG CPLEX Optimization Studio V. 12.7.1.0, a state-of-the-art commercial CP development



(a) FT10 instance solved with Policy 3 (b) FT10 instance solved with Policy 4

(c) FT20 instance solved with Policy 3 (d) FT20 instance solved with Policy 4

Fig. 4. Pareto set approximations: instances FT10 and FT20 using policies 3 and 4

toolkit, and executed on a Core(TM)2 Duo CPU, 3.33 Ghz under Windows 10. Therefore, we used the same running times and the same machine as in [10].

As the Figure 4 shows, our new CP model demonstrates a further improvement over the existing results, especially for the FT20 instance. Relatively to the FT06 instance, the new *energy-aware constraints* do not produce any improvement, most likely because the solutions obtained in [10] are already optimal. The energy-aware propagation rule is less effective on the FT10 instance, as the "short" activity chains that characterize each machine (note that the $jobs/machines$ ratio is 1) do not allow the propagation rule to efficiently evaluate the impact of each state decision on the ed_u variables (see the CP model in Section 3.1). On the FT20 instance, the new CP model provides interesting results, further improving on the solutions obtained in [10] for both the P3 and the P4 policies, confirming that the proposed propagation rule is more effective on instances characterized by a higher $jobs/machines$ ratio. The exact numerical figures related to the Pareto front approximations shown in Figure 4 are reported in Tables 1 and 2, respectively for the P3 and P4 policies. Overall, if we compare policies 3 and 4, we can observe that the latter usually obtains solutions with lower energy consumption. This means that, as expected, the additional possibility of switching the machine to *Stand-by* state is indeed beneficial. For example, in the FT06 instance we were able to reduce the WEC from 126 to 124, while maintaining the optimal makespan of 55. Also, in the solution with the optimal makespan (1165) of FT20, the WEC is reduced from 150 to 120.

Table 1. Pareto set approximations data relative to Figure 4 (Policy P3)

Problem	Pareto Set approximation - set of pairs (MKS, WEC)
FT06	MayEtAl-2015: { (60, 146), (59, 152), (57, 176), (56, 180), (55, 192) }
	OddiEtAl-2017: { (55, 126) }
	CP: { (55, 126) }
FT10	MayEtAl-2015: { (1121, 2708), (1111, 3270), (1097, 3378), (1087, 3430), (1045, 3626), (1034, 3678), (1028, 3792), (1017, 3864), (1016, 4008), (1010, 4188), (998, 4208), (988, 4310), (984, 4570), (982, 4758), (978, 4908), (974, 5840), (963, 5912), (939, 6001), (930, 6013) }
	OddiEtAl-2017: { (1020, 3188), (990, 3658), (980, 3950), (970, 4424), (950, 4446), (940, 5178), (930, 5354) }
	CP: { (1010, 3826), (970, 4252), (950, 4378), (940, 4726) }
FT20	OddiEtAl-2017: { (1185, 0), (1175, 60), (1165, 294) }
	CP: { (1185, 0), (1175, 60), (1165, 150) }

Table 2. Pareto set approximations data relative to Figure 4 (Policy P4)

Problem	Pareto Set approximation - set of pairs (MKS, WEC)
FT06	MayEtAl-2015: { (60, 146), (59, 152), (58, 174), (57, 176), (56, 178), (55, 192) }
	OddiEtAl-2017: { (55, 124) }
	CP: { (55, 124) }
FT10	MayEtAl-2015: { (1121, 2708), (1111, 3268), (1097, 3378), (1087, 3406), (1060, 3512), (1045, 3626), (1034, 3658), (1028, 3792), (1017, 3852), (1010, 3972), (998, 4208), (988, 4310), (984, 4538), (978, 4886), (963, 5182), (951, 5307), (940, 5402), (930, 5786) }
	OddiEtAl-2017: { (1060, 3258), (1040, 3440), (990, 3880), (950, 4356), (945, 4922), (930, 5332) }
	CP: { (1060, 3356), (1010, 3764), (990, 3918), (970, 4226), (960, 4228), (950, 4772), (940, 5150), (930, 5370) }
FT20	OddiEtAl-2017: { (1195, 6), (1185, 18), (1165, 210) }
	CP: { (1235, 0), (1195, 6), (1185, 12), (1178, 18), (1165, 120) }

5 Conclusions

In this paper we have considered a bi-objective optimization in the job shop scheduling problem. We minimise at the same time the makespan and the energy consumption. To this end, we consider an energy model in which each machine can be *Off*, *Stand-by*, *Idle* or *Working*. To solve this complex problem we designed a constraint-programming approach based on a model that exploits energy aware constraints. Our proposal is analyzed and compared against the current state-of-the-art algorithms, obtaining better results. For future work we plan to consider even more realistic energy models. For example if we do not consider the setup and working states together, or also if we consider a flexible environment, i.e. a task can be performed by several machines, each one with different energy consumptions and/or processing times. On the other hand, we can also consider less rich models, for example not quantifying costs but dividing them in costly and non-costly. In this case we expect worse results overall, but it can be a viable approach in some settings where quantifying costs can be difficult.

Bibliography

[1] Apt, K.: Principles of Constraint Programming. Cambridge University Press, New York, NY, USA (2003)

[2] Baker, K.: Introduction to Sequencing and Scheduling. Wiley (1974)

[3] Fisher, H., Thomson, G.L.: Probabilistic learning combinations of local job-shop scheduling rules. In: Muth, J.F., Thomson, G.L. (eds.) Industrial Scheduling, pp. 225–251. Prentice Hall (1963)

[4] González, M.A., Oddi, A., Rasconi, R.: Multi-objective optimization in a job shop with energy costs through hybrid evolutionary techniques. In: Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS-2017). pp. 140–148. AAAI Press, Pittsburgh (2017)

[5] Laborie, P.: Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. Artif. Intell. **143**(2), 151–188 (2003)

[6] Le Pape, C., Baptiste, P., Nuijten, W.: Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems. Springer Science+Business Media, New York, NY, USA (2001)

[7] Liu, Y., Dong, H., Lohse, N., Petrovic, S., Gindy, N.: An investigation into minimising total energy consumption and total weighted tardiness in job shops. Journal of Cleaner Production **65**, 87–96 (2014)

[8] May, G., Stahl, B., Taisch, M., Prabhu, V.: Multi-objective genetic algorithm for energy-efficient job shop scheduling. International Journal of Production Research **53**(23), 7071–7089 (2015)

[9] Miettinen, K.: Nonlinear Multiobjective Optimization. International Series in Operations Research & Management Science, Springer US (2012), https://books.google.it/books?id=bnzjBwAAQBAJ

[10] Oddi, A., Rasconi, R., González, M.: A constraint programming approach for the energy-efficient job shop scheduling problem. In: A., G., Kendall, G., Soon, L., McCollum, B., Seow, H.V. (eds.) Proceedings of the 8th Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2017), 05 - 08 Dec 2017, Kuala Lumpur, Malaysia. pp. 158–172 (2017)

[11] Vilím, P., Barták, R., Cepek, O.: Unary resource constraint with optional activities. In: Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings. pp. 62–76 (2004)

[12] Zhang, R., Chiong, R.: 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 (2016)