Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)

Multi-Objective Optimization in a Job Shop with Energy Costs through Hybrid Evolutionary Techniques

Miguel A. González
Department of Computing
University of Oviedo
Campus de Gijón. 33204 Gijón, Spain
mig@uniovi.es

Angelo Oddi
ISTC-CNR
Via San Martino della Battaglia 44
00185 Rome, Italy
angelo.oddi@istc.cnr.it

Riccardo Rasconi
ISTC-CNR
Via San Martino della Battaglia 44
00185 Rome, Italy
riccardo.rasconi@istc.cnr.it

Abstract

Energy costs are an increasingly important issue in real-world scheduling, for both economic and environmental reasons. This paper deals with a variant of the well-known job shop scheduling problem, where we consider a bi-objective optimization of both the weighted tardiness and the energy costs. To this end, we design a hybrid metaheuristic that combines a genetic algorithm with a novel local search method and a linear programming approach. We also propose an efficient procedure for improving the energy cost of a given schedule. In the experimental study we analyse our proposal and compare it with the state of the art and also with a constraint programming approach, obtaining competitive results.

1 Introduction

The Job Shop Scheduling Problem (JSP) has been a research topic over the last decades due to the fact that it is a simple model of many real production processes.

In particular, the importance of due date related performance criteria has been widely recognized. Meeting due dates is identified as the most important scheduling objective in competitive markets (Wisner and Siferd 1995) (Conner 2009). The total weighted tardiness (TWT) is a due date related objective that is particularly interesting because it can assign different priorities to different operations. Some approaches for the JSP with TWT minimization are the hybrid genetic algorithms proposed in (Essafi, Mati, and Dauzère-Pérès 2008) and (González et al. 2012). Also, in (Kuhpfahl and Bierwirth 2016) some sophisticated and time-consuming neighborhood structures are proposed.

The increasing price of energy, as well as the emission reduction needs, are forcing manufacturing enterprises to put more and more efforts towards the reduction of consumption and the study of energy-saving opportunities and best practices. Existing approaches include the evolutionary algorithms proposed in (Liu et al. 2014) and (Zhang and Chiong 2016), which try to minimize both the weighted tardiness and the energy consumption in a job shop, or the genetic-simulated annealing method of (Dai et al. 2013), that solves a flexible flow shop scheduling problem with energy costs.

Clearly, when the improvement in energy consumption must not be obtained at the cost of losing performance quality in the solutions, we face a bi-objective scheduling problem. There is a growing interest in multi-objective optimization for scheduling and, given its complexity, in the use of metaheuristic techniques to solve them (Dabia et al. 2013).

In this paper we propose an evolutionary algorithm to minimize both the TWT and the energy consumption in a job shop. Our proposal hybridizes several techniques:

- A version of the NSGA-II dominance-based evolutionary algorithm with a mechanism to penalize repeated individuals in the population.
- A multi-objective local search based on hill-climbing.
- A low-polynomial energy post-optimization procedure which attempts to reduce the energy cost of a solution.
- An optimal linear programming approach to further reduce the energy cost of a solution.

The effectiveness of our method is analyzed in the experimental study, and its results are compared with those of the state of the art of the problem considered, which is the NSGA-II proposed in (Liu et al. 2014). We have to remark that, even though our proposal is also based on a NSGA-II algorithm, its crossover operator and replacement strategy are different, and moreover it is hybridized with additional components that lead to a much better performance, as we will see in the experimental study. In this paper we also propose a constraint programming approach and compare its results with those obtained by our evolutionary algorithm.

The remainder of the paper is organized as follows. In Section 2 we formulate the problem. In Section 3 we define our evolutionary algorithm whereas in Section 4 we describe the constraint programming approach. In Section 5 we report the results of the experimental study, and finally Section 6 summarizes the main conclusions of this paper.

2 Problem formulation

In the job shop scheduling problem, a set of N jobs, $J = \{J_1, \ldots, J_N\}$, are to be processed on a set of M machines or resources, $R = \{R_1, \ldots, R_M\}$. Each job J_i consists of a sequence of n_i operations $(\theta_{i1}, \ldots, \theta_{in_i})$, where each operation requires the uninterrupted use of a given machine during all its processing time. The objective is to minimize some objective functions subject to the following constraints: (i) the sequence of machines for each job is prescribed, and (ii)

each machine can process at most one operation at a time. Jobs may also have a due date, that is, a time before which all its operations should be completed, and a weight, which represents its relevance. In order to simplify expressions, instead of using θ_{ij}, in the following we are denoting operations by a single letter whenever possible. We denote by:

- Ω the set of operations
- d_i the due-date of job J_i
- w_i the weight of job J_i
- P_k^{idle} the idle power level of machine R_k
- p_u the processing time of operation u
- s_u the starting time of operation u that needs to be determined

The JSP has two binary constraints: precedence and capacity. Precedence constraints, defined by the sequential routings of the operations within a job, translate into linear inequalities of the type: $s_u + p_u \leq s_v$, where v is the next operation to u in the job sequence. Capacity constraints that restrict the use of each machine to only one operation at a time translate into disjunctive constraints of the form: $(s_u + p_u \leq s_v) \vee (s_v + p_v \leq s_u)$, where u and v are operations requiring the same machine.

The objective here is to obtain a feasible schedule, i.e. a starting time for each one of the operations such that all constraints are satisfied. In the following, given a feasible schedule, PJ_v and SJ_v denote respectively the predecessor and successor of v in the job sequence, and PM_v and SM_v the predecessor and successor of v in its machine sequence.

We are minimizing two objective functions: the total weighted tardiness and the energy consumption. The TWT is defined as follows

$$\sum_{i=1,\ldots,N} w_i T_i \tag{1}$$

T_i is the tardiness of the job i, given by $T_i = \max\{C_i - d_i, 0\}$, where C_i is the completion time of job i.

The energy consumption model is taken from (Liu et al. 2014), where it is supposed that the machines cannot be turned off when idle. In that case it is proven that the objective to reduce the total electricity consumption of a job shop can be converted to reduce the total non-processing energy (NPE), i.e. the amount of time a machine is on and not executing a job. Notice that each machine must process a fixed set of operations, all of them having fixed durations, and so for any schedule the processing energy must be equal.

Hence, the objective function can be set as the sum of all the NPE consumed by all the machines in a job shop to carry out a given job schedule. Then, the total NPE is defined as

$$\sum_{k=1,\ldots,M} [P_k^{idle} \times (s_{\omega_k} + p_{\omega_k} - s_{\alpha_k} - \sum_{u \in M_k} p_u)] \tag{2}$$

where α_k and ω_k are the first and last operations respectively on machine R_k in the considered schedule, and M_k is the set of operations that must be executed in machine R_k.

The TWT is a regular performance measure (Baker 1974), which means that its value can be increased only by increasing at least one of the completion times in the schedule. To minimize a regular measure, it is sufficient to consider "left-shift schedules"; i.e. schedules built from a partial ordering of the operations, so that each operation starts as soon as possible after all the preceding operations in the partial ordering. On the contrary, the NPE is a non-regular performance measure because it can be increased if we are able to decrease the starting time of the first operation of a machine while maintaining all other operations intact.

3 The multi-objective evolutionary algorithm

In general, for a minimization problem with $f_i, i = 1, \ldots, n$ objective functions, a solution S is said to be *dominated* by another solution S', denoted $S' \succ S$, 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 $f_i(S') < f_i(S)$. Our goal will then be to find non-dominated solutions to our problem with respect to TWT and NPE. To achieve this, we propose a dominance-based hybrid method, combining a multi-objective evolutionary algorithm with a multi-objective hill climbing local search and a linear programming approach.

Some papers have already considered minimizing non-regular objectives in a job shop. As an example, in (Brandimarte and Maiocco 1999) the authors tackle a single-objective case and propose to decompose the overall problem into sequencing and timing subproblems. We follow a similar approach, in the sense that we represent the solutions as permutations in the genetic algorithm and in the local search in order to solve the sequencing subproblem. To solve the timing subproblem we introduce a low-polynomial energy post-optimization procedure when evaluating each solution, and also a more computationally expensive optimal linear programming approach to further improve the NPE of the final set of non-dominated solutions.

3.1 Genetic algorithm

Our proposal is based on the well-known NSGA-II template for a dominance-based evolutionary algorithm (Deb et al. 2002). An initial population Pop_0 of size $popSize$ is randomly created and evaluated and then the algorithm iterates over $numGen$ generations, keeping a set of non-dominated solutions. At each generation i a new population $\mathrm{Off}(Pop_i)$ is built from the current one Pop_i by applying selection, crossover and mutation, and finally a replacement strategy is applied to obtain the next generation Pop_{i+1}.

Representation and evaluation of chromosomes Solutions are codified into chromosomes using permutations with repetition, as introduced in (Bierwirth 1995) for the JSP. This is a permutation of the set of operations, each being represented by its job number, which represents a linear ordering compatible with precedence constraints. For example, if we have a problem instance with 3 jobs: $J_1 = \{\theta_{11}, \theta_{12}\}$, $J_2 = \{\theta_{21}, \theta_{22}, \theta_{23}, \theta_{24}\}$, $J_3 = \{\theta_{31}, \theta_{32}, \theta_{33}\}$, then the ordering of operations $\pi = \{\theta_{21}, \theta_{11}, \theta_{22}, \theta_{31}, \theta_{23}, \theta_{32}, \theta_{33}, \theta_{24}, \theta_{12}\}$ is represented by the chromosome $v = (2\ 1\ 2\ 3\ 2\ 3\ 3\ 2\ 1)$.

A given chromosome is evaluated by generating an associated schedule and then computing its TWT and NPE. To this end, we use an insertion strategy following the sequence

given by the chromosome. Each operation is scheduled at the earliest possible starting time such that all constraint are met, without moving previously scheduled operations.

Energy post-optimization procedure The procedure described for evaluating a chromosome would be able to produce a Pareto optimal schedule if the objective functions were regular. However, the NPE is a non-regular objective function, and therefore scheduling each operation as soon as possible may not be the best option. Notice that we can reduce the NPE of a machine R_k by delaying the starting time of its first operation (s_{α_k}) without increasing the starting time of its last operation (s_{ω_k}). Of course, it can also be reduced by decreasing s_{ω_k} without increasing s_{α_k}, but this approach is not possible in our method, since operations are initially scheduled as soon as possible.

We propose to use an additional energy post-optimization procedure that, given a schedule, it maintains the operation ordering and tries to reduce its NPE while not increasing its TWT. Basically, the idea is to delay all the operations of each machine as much as possible, with the exception of the last one, increasing the tardiness of none of the jobs. Of course, we always take into account the precedence constraints between the operations of a job. The procedure is detailed in Algorithm 1. Notice that, by not increasing the starting times of the last operation of each machine, we ensure that the resulting NPE after applying the procedure is lower than or equal to the original one. Furthermore, the TWT of the resulting solution is not increased either, as the completion time of the last operation of a job may only be delayed if it is lower than the due date, and in this case it is at most delayed up to this due date.

This procedure is executed inside the solution evaluation method, just after the schedule is built. Therefore, it is applied to evaluate every chromosome generated in the genetic algorithm and every neighbor considered in the local search.

When adding this procedure to the scheduler, the running time of the evolutionary algorithm increases between 10% and 25%, but at the same time the results significantly improve, as we will see in Section 5.

Genetic operators The selection phase selects the chromosomes that will undergo crossover and mutation, and is based on a tournament strategy. We select $tSize$ chromosomes at random and choose the best one to be the first parent, according to non-domination rank and crowding distance (see next section). Then we select a second parent in the same way, and the crossover operator is applied with probability $crProb$ to obtain two offspring solutions.

For chromosome mating we have considered the Job Order Crossover (JOX) (Bierwirth 1995). Given two parents, JOX selects a random subset of jobs and copies their genes to the offspring in the same positions as they are in the first parent, then the remaining genes are taken from the second parent so as to maintain their relative ordering. A second offspring is generated reversing the role of the parents. Then, each offspring is mutated with probability $mutProb$ using the swap mutation operator, which swaps two positions of the chromosome chosen at random.

Algorithm 1 The energy post-optimization procedure

Require: A problem instance I and a feasible schedule (i.e. an ordering O and a set of starting times s)
  $k \leftarrow |\Omega|$
  while $k \geq 1$ **do**
    $a = O[k]$;
    if a is the last operation processed in a machine **then**
      $s'_a = s_a$;
    else
      if a is the last operation of its job **then**
        $j \leftarrow$ job of operation a;
        $s'_a = \min\{\max\{d_j, s_a + p_a\}, s'_{SM_a}\} - p_a$;
      else
        $s'_a = \min\{s'_{SJ_a}, s'_{SM_a}\} - p_a$;
      end if
    end if
    $k \leftarrow k - 1$;
  end while
  return A new set of starting times s' for the ordering O

Replacement strategy The replacement strategy establishes how population Pop_i and population $\mathrm{Off}(Pop_i)$ that results from applying selection, crossover and mutation to Pop_i are combined to generate the new population Pop_{i+1}. Here we adopt a strategy based on the non-dominated sorting approach with diversity preservation from (Deb et al. 2002). Initially, for each individual j in the pool $Pop_i \cup \mathrm{Off}(Pop_i)$ a non-domination rank ($rank(j)$) and a crowding distance ($dist(j)$) are calculated. The former sorts the pool into different non-domination levels while the latter estimates the density of solutions in the area of the non-domination level where the individual lies. Population Pop_{i+1} is then formed by the best $popSize$ individuals from the pool $Pop_i \cup \mathrm{Off}(Pop_i)$ according to the lexicographical order defined by $(rank, dist)$. That is, solutions belonging to a lower (better) non-domination rank are preferred and, between two solutions in the same non-dominance level, we prefer the solution located in the less crowded region.

We have added an additional step in order to improve the diversity. Specifically, we propose to start by removing from the pool of individuals $Pop_i \cup \mathrm{Off}(Pop_i)$ those which are repeated, in the sense that there exists in the pool at least another individual having identical values for all objective functions. Only after this elimination is the above strategy based on $(rank, dist)$ applied. If such elimination causes the pool to contain less than $popSize$ individuals, all the non-repeated individuals pass onto the next generation Pop_{i+1}, which is then completed with some of the repeated individuals by recursively using the same strategy.

3.2 Local search

Local search is often used in combination with other metaheuristics in such a way that the local search provides exploitation while the other metaheuristic provides exploration. The main difficulty when designing multi-objective memetic algorithms is the implementation of the local search, which essentially is a single-objective optimization

technique. According to (Liefooghe et al. 2012), the number of multi-objective local search algorithms proposed so far is still reduced. There are some Pareto-based local searchers in the literature, as for example PAES (Pareto Archived Evolution Strategy) proposed in (Knowles and Corne 2000) or the Pareto Local Search proposed in (Paquete, Schiavinotto, and Stützle 2007). One inconvenience of these local searchers is that they are too computationally costly to be combined with a genetic algorithm. We propose a less time-consuming local search procedure that provides a single output solution, what is called "one-point iteration" in (Lara et al. 2010).

Another issue when applying local search to a multi-objective setting is how to establish a selection criterion for the best neighbor, since the dominance relation $\succ$ defines a partial order. For instance, in (Ishibuchi et al. 2009) and (Jaszkiewicz 2003) the authors propose to scalarise the objective function vector to guide the search. Other authors propose instead to define acceptance criteria based on dominance, as for example in (Knowles and Corne 2000).

In this paper we propose a fast and efficient local search based on hill climbing in which the selection of the neighbor is based on dominance but, at the same time, it considers the solutions in the current non-dominated set of solutions of the genetic algorithm. Hence, a number of neighbors may be chosen, even if they do not dominate the current solution, as long as they are actually interesting. Our approach starts with a solution provided by the genetic algorithm, and generates neighbors of the solution one by one. Each one is evaluated with the scheduler and the energy post-optimization procedure described in Section 3.1, until we find one neighbor that fulfills one of the following conditions:

1. The neighbor dominates the current solution.
2. The neighbor would enter in the current set of non-dominated solutions of the genetic algorithm (i.e. no solution of the population dominates the neighbor and also no solution has the exact same fitness values as the neighbor), while the current solution would not enter.

As soon as one such neighbor is found, the procedure swaps the current solution for the newly found solution and repeats the process. On the other hand, if no such neighbor exists the procedure ends, returning the current solution.

Notice that the second condition is very useful, allowing us to select very interesting neighbors that may not dominate the current solution. However, if the current solution would already enter in the non-dominated set of solutions of the genetic algorithm, we opt to choose dominating neighbors only, in order to not lose the obtained solution.

The local search is detailed in Algorithm 2. It is actually not very time consuming, so it may be applied to all initial chromosomes and to all generated offsprings of the genetic algorithm. Notice that this would not be reasonable for other alternatives such as the Pareto Local Search (Paquete, Schiavinotto, and Stützle 2007), given their greater computational load. After the local search is applied, the chromosome is rebuilt from the improved schedule obtained, so its characteristics can be transferred to subsequent offsprings. This effect is known as Lamarckian evolution.

Algorithm 2 Multi-objective hill climbing local search

Require: A problem instance I and a feasible schedule S
  $S' \leftarrow S$; $continue \leftarrow True$;
  while $continue = True$ **do**
    $NeighborSelected \leftarrow False$;
    $N(S') \leftarrow$ neighborhood of S';
    $k \leftarrow 1$;
    while $NeighborSelected = False$ and $k \leq |N(S')|$ **do**
      $S'' \leftarrow N(S')[k]$;
      Evaluate S'';
      if S'' dominates S', or S'' would enter in the current set of non-dominated solutions of the genetic algorithm and S' would not **then**
        $NeighborSelected \leftarrow True$;
      end if
      $k \leftarrow k + 1$;
    end while
    if $NeighborSelected = False$ **then**
      $continue \leftarrow False$
    else
      $S' \leftarrow S''$
    end if
  end while
  return The (hopefully) improved solution S' for I;

Neighborhood structure Several neighborhoods have been proposed in the literature for the JSP, and most of them rely on the concepts of critical path and critical block. These concepts are based on the disjunctive graph representation of the problem. Here we use a similar representation as other papers in the literature (see (Essafi, Mati, and Dauzère-Pérès 2008; González et al. 2012; Kuhpfahl and Bierwirth 2016)). A critical path is defined as a largest cost path from the start node to an end node (when minimizing TWT the graph has one end node for each job). The length of each path determines the contribution of that particular job to the solution cost. A critical block is a maximal subsequence of consecutive operations in a critical path requiring the same machine, whereas a critical arc is an arc inside a critical block.

Here we adopt the neighborhood structure initially proposed in (Van Laarhoven, Aarts, and Lenstra 1992), based on reversing a single critical arc. It has some nice properties, in particular, it always generates feasible neighbors, avoiding the need of repairing procedures. In TWT minimization the cost of a solution can be given by up to N critical paths, one for each job that ends after its due date. In order to limit the computational burden of the local search we opted to reduce the number of neighbors, considering only the critical path of the job that contributes the most to the TWT of the schedule. A similar idea was proposed in (González et al. 2012). Although this structure is designed to minimize the TWT, we have empirically seen that most neighbors that improve the TWT also improve the NPE as well.

3.3 The linear programming approach

The solution returned by the energy optimization procedure described in Section 3.1 could be further improved in the following way: if we keep the processing ordering of the operations on the machines, delaying the starting time of the last operation of some machine may allow the first operation of another machine to be delayed as well, giving rise to a reduction in the overall energy consumption. Of course, checking for all these possibilities is computationally expensive and so such procedure could not be applied after each chromosome evaluation. However, it can be applied, for example, to the solutions in the Pareto set approximation returned by the memetic algorithm. To this end, given the problem definition of Section 2 and an input solution S, we consider the following relaxed Linear Programming (LP) problem.

$$min \sum_{k=1,..,M} [P_k^{idle} \times (s_{\omega_k} + p_{\omega_k} - s_{\alpha_k} - \sum_{u \in M_k} p_u)]$$

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

$$s_v + p_v \leq s_{SM_v}$$

$$v \in M_k \setminus \{\omega_k\}, k = 1, .., M \tag{3b}$$

$$0 \leq s_{\theta_{i1}} \quad i = 1, .., N \tag{3c}$$

$$s_{\theta_{in_i}} + p_{\theta_{in_i}} \leq \max\{C_i, d_i\} \quad i = 1, .., N \tag{3d}$$

Decision variables are the starting times of the operations s_v with $v \in \Omega$. Constraints (3a) represent the linear orderings imposed on the set of operations Ω by the jobs J, note that they hold for each operation $v \in \Omega$ except when v is the last operation of a job J_i. The processing orderings on the machines in S are represented by constraints (3b). Constraints (3c) impose to the first operation θ_{i1} of each job J_i to start after the reference value 0. Finally, constraints (3d), imposed on the end times of the last operations θ_{in_i} of each job J_i, guarantee that the final value of the TWT objective is less than or equal to that of the input solution S. As it is easy to verify, all the imposed temporal constraints are of the kind $x - y \leq c$. Hence, in accordance with (Papadimitriou and Steiglitz 1982; Sakkout and Wallace 2000) the coefficient matrix of the above LP is *totally unimodular* (TU), and therefore all the optimal solutions of the LP problem remain discrete values and they provide the optimal *NPE* given the processing ordering established by the input solution S. Similar considerations are proposed in (Brandimarte and Maiocco 1999), where the optimal timing problem for non-regular single objectives in a job-shop are reduced to a minimum cost flow problem. We propose to apply this linear programming approach in all solutions of the Pareto front obtained in the last generation of the memetic algorithm, in order to further improve its final results.

4 A Constraint Programming approach

Constraint Programming (CP) is a declarative programming paradigm (Apt 2003) suitable for solving constraint satisfaction and optimization problems. 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 variables domains, and search.

Constraint Programming is particularly suited for solving scheduling problems where the decision variables correspond to the problem operations. In particular, each operation variable u is characterized at least by two features: s_u representing its start time, and p_u representing its duration. For scheduling problems, a number of different *global constraints* have been developed, the most important being the `unary-resource` constraint (Vilím, Barták, and Cepek 2004) for modelling simple machines, the `cumulative` resource constraint (Le Pape, Baptiste, and Nuijten 2001) for modelling cumulative resources (e.g., a pool of workers), and the `reservoir` (Laborie 2003) 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 probably inconsistent assignments of operation start-time variables.

Our CP model takes into account the non-regularity of the *NPE* objective function by introducing a set additional decision variables representing the switch *ON/OFF* events in the set of machines M_k. Given the problem defined in Section 2, we consider the following model.

$$\min \sum_{k=1,...,M} [P_k^{idle} \times (s_{\omega_k} + p_{\omega_k} - s_{\alpha_k} - \sum_{u \in M_k} p_u)]$$

$$\text{s.t.}: \quad s_v + p_v \leq s_{SJ_v} \quad v \in \Omega \tag{4a}$$

$$\text{span}(\mathit{OnOff}_k, M_k) \quad k = 1, 2, .., M \tag{4b}$$

$$\texttt{unary-constraint(}M_k\texttt{)} \quad k = 1, .., M \tag{4c}$$

Decision variables are the starting times of the operations $s_v \in \Omega$ extended with the start times s_{OnOff_k} of the operations OnOff_k, representing the first instant when machine k is turned on, such that, it remains in this state for its entire duration p_{OnOff_k}. We assume that each decision variable s_a ranges in the interval $[0, H - p_a]$.

The objective function to minimize represents the *NPE* value (the TWT objective function could be expressed analogously). Constraints (4a) represent the linear orderings imposed on the set of operations Ω by the set of jobs J. Note that if ω_i is the last operation of a job J_i, then its successor is the horizon point $SJ_{\omega_i} = H$, $i = 1, 2, .., N$. Constraints (4b) impose to the set M_k of operations requiring machine k to be contained in the *spanning* operations OnOff_k, $k = 1, 2, \ldots, M$. More specifically, for each operation $u \in M_k$, the following constraints $s_{\mathit{OnOff}_k} \leq s_u$ and $s_u + p_u \leq s_{\mathit{OnOff}_k} + p_{\mathit{OnOff}_k}$ hold. Finally, (4c) represents the non-overlapping constraints imposed by the machines M_k through the global constraints `unary-constraint(`M_k`)`. We have to remark that the s_{OnOff_k} decision variables are not really necessary, and the model definition remains perfectly feasible and solvable without them. However, in a preliminary study we have observed that their addition is beneficial and helps the solver.

A well-studied multi-objective optimization method to generate the whole Pareto front is the ϵ-constraint method (Miettinen 2012). It works by choosing one objective function as the only objective and properly constraining the remaining objective functions during the optimization process.

Algorithm 3 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**
    $S \leftarrow$ CP($\mathbf{f}$, ϵ);
    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

Through a systematic variation of the constraint bounds, different elements of the Pareto front can be obtained.

Algorithm 3 presents the ϵ-constraint method for the case of a bi-criterion objective function $\mathbf{f} = (f^{(1)}, f^{(2)})$. The algorithm 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 single-objective problems (the CP() procedure corresponding to the constraint programming model previously described). 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 set P, then S is inserted in P, and all the solutions possibly dominated by S are removed from P. The rationale behind this method is to iteratively tighten the constraint bound by a pre-defined constant δ at each step of the solving cycle.

5 Experimental results

In this section we provide an empirical evaluation of the proposed algorithms. Experiments were made on instances available in the literature (Liu et al. 2014). Specifically we considered one instance that was generated based on the well-known FT10 instance of the JSP, of size 10×10, adding due dates, job weights and the idle power consumption of each machine. The due date of each job J_i was assigned using the expression $d_i = k \times \sum_{j=1}^{n_i} p_{ij}$, where k is a parameter that controls the tightness of the due dates, being 1.5, 1.6, 1.7 and 1.8 in our work. Therefore, there are 4 instances in all (see (Liu et al. 2014) for more details).

The memetic algorithm is implemented in C++ using a single thread, while the Linear Programming (LP) step and the Constraint Programming (CP) approach are implemented using IBM CPLEX Optimization Studio 12.6. Our experiments were carried out on a Intel Core i5-2450M CPU at 2.5 GHz with 4 GB of RAM, running on Windows 10 Pro.

We have conducted a preliminary parametric analysis to set some of the parameters of our approach. Table 1 shows a summary of the tested values, indicating in bold the configuration that achieved the best average results. To do a comparison as fair as possible, we modified $numGen$ accordingly so that the running time is reasonable and similar for all configurations tested: about 4 minutes per run. Using this configuration, and setting $numGen = 2000$, the convergence pattern is appropriate and the time spent in each part of the algorithm is roughly 73% for the local search, 27% for the genetic algorithm, and less than 1% for the linear programming step. Even if the time spent by the LP step is less than 1 second, it would be too computationally expensive to apply it in every generation of the algorithm.

Table 1: Values tested in the parameter tuning. Bold values indicate the best configuration found.

Parameter	Values tested
$popSize$	500, **1000**, 2000
$tSize$	**2**, 4, 8, 16
$crProb$	0.6, 0.8, **1.0**
$mutProb$	0, 0.1, **0.2**, 0.3

Figures 1, 2, 3 and 4 show the Pareto fronts obtained with our methods and compared with those obtained in (Liu et al. 2014), for $k = 1.5, 1.6, 1.7$ and 1.8. Our memetic algorithm enhanced with the energy Post-Optimization procedure and the final Linear Programming step is labelled $Memetic + PO + LP$ in all figures.

The proposal of (Liu et al. 2014) (labelled LIU) is a standard NSGA-II algorithm using OOX crossover operator and swap mutation. The crossover probability is set at 1.0 and the mutation probability at 0.6. The population size varies between 800 and 1000 depending on the instance, and the total number of generations vary between 25000 and 40000. As the authors do not report the computational time, we have implemented a version of their method and concluded that their running time is considerably larger than that of our approach: about 15 minutes per run. To have a reference value for weighted tardiness, in (Liu et al. 2014) the authors also report results with the LEKIN software, using the Shifting Bottleneck and Local Search heuristics. These are used to perform a single-objective optimization of the weighted tardiness, and so the result in this case is a single solution instead of a Pareto front. It is labelled $LEKIN$ in all figures.

About the CP approach, we generate an approximation of the Pareto set by using the *ϵ-constraint* algorithm (see Algorithm 3) by running twice the procedure: in the first run we optimize the *NPE* while in the second run we optimize the *TWT*. For each run we set a total of 10 solving cycles and a total bound of 1000 seconds on the CPU time, as this value is close to the 15 minutes above estimated for the LIU algorithm. At each cycle we leverage on the random nature of the CP Optimizer algorithm - a Large Neighborhood Search strategy - and run 10 times the solver, each with a CPU time bound of 10 seconds. About the intervals $[f^{(2)}_{min}, f^{(2)}_{max}]$ on the objective functions, we used as basis the values from (Liu et al. 2014). In particular, when we optimize the *TWT* objective, the interval of *NPE* values is $[60, 200]$ for each value of k. Whereas, when we optimize *NPE*, according to the values of $k = 1.5, 1.6, 1.7$, and 1.8, the interval of *TWT* values are $[1500, 3500]$, $[1000, 3000]$, $[500, 2500]$, and

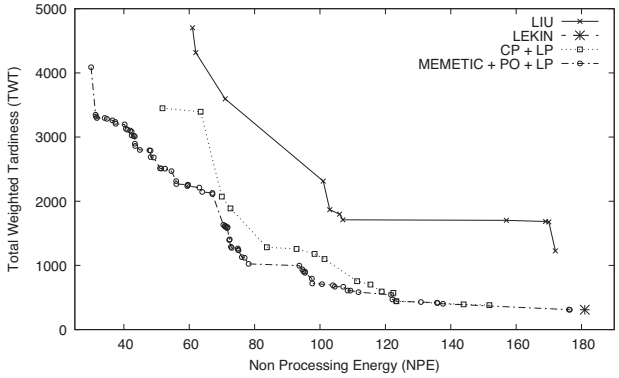


Figure 1: Pareto fronts obtained with $k = 1.5$

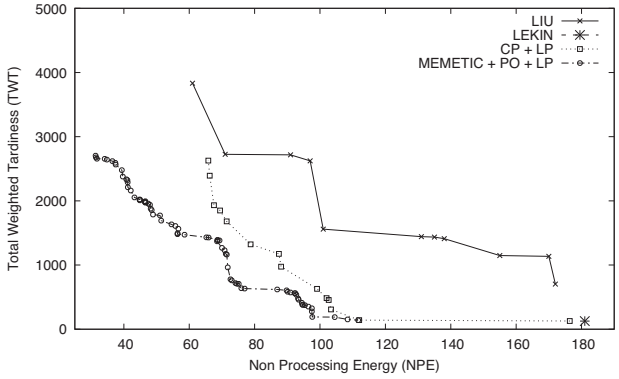


Figure 2: Pareto fronts obtained with $k = 1.6$

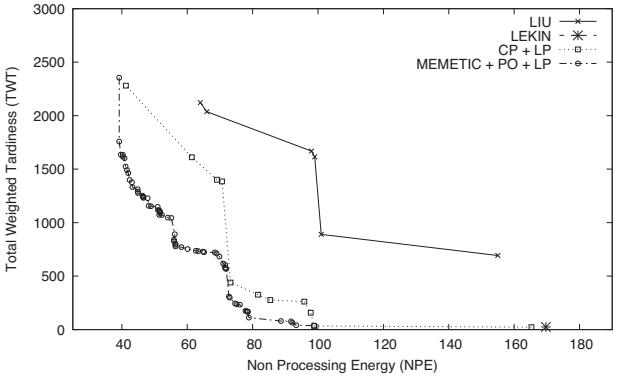


Figure 3: Pareto fronts obtained with $k = 1.7$

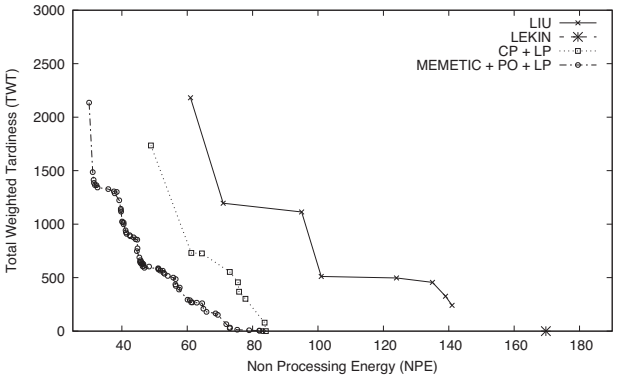


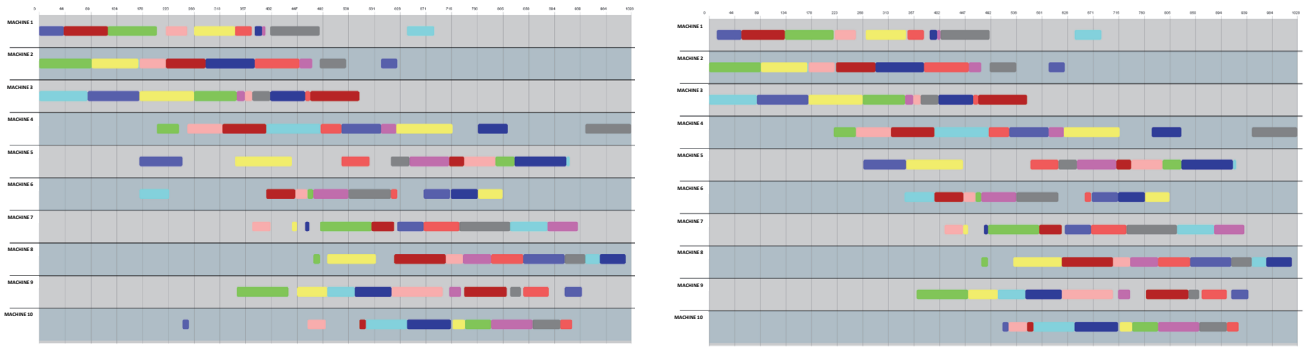
Figure 4: Pareto fronts obtained with $k = 1.8$

$[0, 2000]$ respectively. Afterwards, we merge the two approximations of the Pareto sets into a single one. Finally, we apply the Linear Programming step described in Section 3.3 to the set of solutions obtained with the CP algorithm. It is worth noting that we tested additional ways to generate the Pareto set within the same total CPU time: a run with a single objective and/or a single run at each solving step of the ϵ-*constraint* algorithm. In all cases, the chosen method (labelled ***CP + LP*** in all figures) gave the best performance.

All plots demonstrate that the CP-based algorithm clearly outperforms LIU, whereas the hybrid evolutionary algorithm further improves over the CP performance. Hence, even if the ϵ-*constraint* algorithm is a basic strategy to generate a Pareto set, the set of results generated with a state-of-the-art CP solver is clearly a further confirmation of the effectiveness of the proposed evolutionary approach.

The hypervolume values (Zitzler and Thiele 1998) of all the Pareto fronts shown in Figures 1, 2, 3 and 4 are summarized in Table 2, computed with the reference point equal to $(200, 5000)$ for all instances and all k values. In addition, Table 2 reports the hypervolume values obtained by some variants of our evolutionary algorithm, stripping it of some of its characteristics: $Genetic$ (only the plain genetic algorithm), $Memetic$ (genetic algorithm with local search), $Genetic + PO$ and $Memetic + PO$ (adding the energy post-optimization procedure to the scheduler), $Genetic+PO+LP$ (further adding the linear programming step to $Genetic + PO$), and finally CP (running the CP approach without the linear programming step). We adjusted the stopping condition of our hybrid evolutionary methods so that running times are similar, in order to achieve a comparison as fair as possible. For instance, for $Memetic$ we set $numGen = 2500$, for $Genetic$ we set $numGen = 8000$ and for $Genetic + PO$ and $Genetic + PO + LP$ we set $numGen = 7200$.

The hypervolume values confirm the superiority of our evolutionary algorithm over the results obtained in (Liu et al. 2014). In addition, we would like to stress how the post-optimization procedure introduced in this paper represents a further and remarkable performance boost (compare $Genetic$ with $Genetic+PO$, or $Memetic$ with $Memetic+PO$). In fact, it should be underscored at this point that the results obtained with the post-optimization step are indeed extremely close to those obtained by applying the optimal Linear Programming approach (the hypervolume values are very close in all cases), thus proving the effectiveness of the post-optimization (PO) procedure. However, despite the total hypervolume improvement is very small, the Linear Programming procedure was able to improve a significant number of solutions in all cases. For example, if we compare



(a) Before post-optimization ($TWT = 3347$, $NPE = 91.98$ KWh) (b) After post-optimization ($TWT = 3347$, $NPE = 31.35$ KWh)

Figure 5: Improvement of the NPE value as a consequence of the application of the post-optimization procedure (Algorithm 1).

Table 2: Hypervolumes computed for all procedures.

	k=1.5	k=1.6	k=1.7	k=1.8
Genetic	0.6272	0.6801	0.7237	0.7842
Genetic + PO	0.6377	0.7266	0.7556	0.7842
Genetic + PO + LP	0.6380	0.7272	0.7564	0.7868
Memetic	0.6238	0.6964	0.7620	0.7941
Memetic + PO	0.6654	0.7364	0.7635	0.8203
Memetic + PO + LP	**0.6656**	**0.7366**	**0.7637**	**0.8206**
CP	0.5732	0.6053	0.7173	0.7212
CP + LP	0.5803	0.6067	0.7180	0.7214
Liu	0.3862	0.4553	0.5264	0.6038

$Memetic+PO+LP$ with $Memetic+PO$, it improved 23 solutions on 76 in the $k = 1.5$ case, 31 solutions on 74 in the $k = 1.6$ case, 31 solutions on 68 in the $k = 1.7$ case, and 35 solutions on 73 with $k = 1.8$. Therefore, even if the Linear Programming step does not appear to be extremely important, it also contributes to provide further improvements.

The hybridization with local search clearly improves the performance: $Memetic$ is better than $Genetic$ in most cases, whereas $Memetic+PO$ and $Memetic+PO+LP$ are always better than its counterparts $Genetic + PO$ and $Genetic + PO + LP$. We can also observe that our simple $Genetic$ algorithm (even without local search, PO and LP) is much better than Liu. The main reason is the procedure for eliminating repeated individuals to create each generation, which dramatically improves the diversity of the population. Our crossover operator (JOX) also represents an slight improvement with respect to the OOX used in Liu.

Finally, Figure 5 shows two solutions, respectively *before* and *after* the application of the post-optimization algorithm presented in Section 3.1, in the $k = 1.5$ case (the behavior is similar with other values of k). The TWT is the same, while the *NPE* value is significantly improved by introducing delays on the start times of some operations (the most evident delays are those related to machines R_5, R_6, and R_{10}).

6 Conclusions

We have considered the problem of minimizing both the total weighted tardiness and the energy consumption in a job shop. To this end, we have proposed a multi-objective approach that hybridizes a NSGA-II based evolutionary algorithm with a multi-objective local search. To optimize the energy consumption we have designed two methods: a fast, low-polynomial procedure to be included in the chromosome evaluation algorithm, and a linear programming approach which is more costly and applied only to the final set of non-dominated solutions, to further improve them. In the experimental study we have proven the efficiency of the proposed energy-optimization procedures and we have seen that our approach improves the results of the state of the art, and also those obtained by a constraint programming approach.

In our opinion, the remarkable performance of our algorithm is due to the combination of the diversification provided by the NSGA-II combined with the intensification provided by the local search. The fast local search and the reduction of the neighborhood allowed us to apply it to every solution in a reasonable computational time. Additionally, the proposed energy optimization methods significantly improved the quality of the solutions.

For future work we will try different multi-objective algorithms, as for example MOEA-D or PAES. Additionally, we plan to design a benchmark with more instances. Another very interesting possibility is to consider more realistic energy consumption models; for example models that consider shifting energy costs (Grimes et al. 2014), models that allow varying the energy consumed by varying the processing mode of operations (Zhang and Chiong 2016), or models where the machines can be Turn off/Turn on (Mouzon, Yildirim, and Twomey 2007).

Acknowledgements

We thank Ying Liu for providing us with the detailed results of his work. 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”.

References

Apt, K. 2003. *Principles of Constraint Programming*. New York, NY, USA: Cambridge University Press.

Baker, K. 1974. *Introduction to Sequencing and Scheduling*. Wiley.

Bierwirth, C. 1995. A generalized permutation approach to jobshop scheduling with genetic algorithms. *OR Spectrum* 17:87–92.

Brandimarte, P., and Maiocco, M. 1999. Job shop scheduling with a non-regular objective: a comparison of neighbourhood structures based on a sequencing/timing decomposition. *International Journal of Production Research* 37(8):1697–1715.

Conner, G. 2009. 10 questions. *Manufacturing Engineering Magazine* 93–99.

Dabia, S.; Talbi, E.-G.; van Woensel, T.; and De Kok, T. 2013. Approximating multi-objective scheduling problems. *Computers & Operations Research* 40:1165–1175.

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

Deb, K.; Pratap, A.; Agarwal, S.; and Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2):182–197.

Essafi, I.; Mati, Y.; and Dauzère-Pérès, S. 2008. A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. *Computers & Operations Research* 35:2599–2616.

González, M. A.; González-Rodríguez, I.; Vela, C.; and Varela, R. 2012. An efficient hybrid evolutionary algorithm for scheduling with setup times and weighted tardiness minimization. *Soft Computing* 16:2097–2113.

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.

Ishibuchi, H.; Hitotsuyanagi, Y.; Tsukamoto, N.; and Nojima, Y. 2009. Use of biased neighborhood structures in multiobjective memetic algorithms. *Soft Computing* 13(8–9):795–810.

Jaszkiewicz, A. 2003. Do multiple-objective metaheuristics deliver on their promises? A computational experiment on the set-covering problem. *IEEE Transactions on Evolutionary Computation* 7(2):133–143.

Knowles, J. D., and Corne, D. W. 2000. Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation* 8(2):149–172.

Kuhpfahl, J., and Bierwirth, C. 2016. A study on local search neighborhoods for the job shop scheduling problem with total weighted tardiness objective. *Computers & Operations Research* 66:44–57.

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

Lara, A.; Sánchez, G.; Coello Coello, C. A.; and Schütze, O. 2010. HCS: A new local search strategy for memetic multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 14(1):112–132.

Le Pape, C.; Baptiste, P.; and Nuijten, W. 2001. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. New York, NY, USA: Springer Science+Business Media.

Liefooghe, A.; Humeau, J.; Mesmoudi, S.; Jourdan, L.; and Talbi, E.-G. 2012. On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. *Journal of Heuristics* 18(2):317–352.

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

Miettinen, K. 2012. *Nonlinear Multiobjective Optimization*. International Series in Operations Research & Management Science. Springer US.

Mouzon, G.; Yildirim, M. B.; and Twomey, J. 2007. Operational methods for minimization of energy consumption of manufacturing equipment. *International Journal of Production Research* 45(18–19):4247–4271.

Papadimitriou, C., and Steiglitz, K. 1982. *Combinatorial Optimization: Algorithms and Complexity*. Dover Books on Computer Science. Dover Publications.

Paquete, L.; Schiavinotto, T.; and Stützle, T. 2007. On local optima in multiobjective combinatorial optimization problems. *Annals of Operations Research* 156:83–97.

Sakkout, H., and Wallace, M. 2000. Probe backtrack search for minimal perturbation in dynamic scheduling. *Constraints* 5(4):359–388.

Van Laarhoven, P.; Aarts, E.; and Lenstra, K. 1992. Job shop scheduling by simulated annealing. *Operations Research* 40:113–125.

Vilím, P.; Barták, R.; and Cepek, O. 2004. 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*, 62–76.

Wisner, J., and Siferd, S. 1995. A survey of us manufacturing practices in make-to-order machine shops. *Production and Inventory Management Journal* 1:1–7.

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.

Zitzler, E., and Thiele, L. 1998. Multiobjective optimization using evolutionary algorithms — a comparative case study. In *Parallel Problem Solving from Nature — PPSN V Proceedings*. Springer Berlin Heidelberg. 292–301.