The Knowledge Engineering Review, Vol. 31:5, 475–485.
doi:10.1017/S026988891600031X

A metaheuristic technique for energy-efficiency in job-shop scheduling

JOAN ESCAMILLA[1], MIGUEL A. SALIDO[1], ADRIANA GIRET[2] and FEDERICO BARBER[1]

[1]*Instituto de Automática e Informática Industrial, Universidad Politécnica de Valencia, Camino de vera s/n, 46022 Valencia, Spain; e-mail: jescamilla@dsic.upv.es*, msalido@dsic.upv.es, fbarber@dsic.upv.es;
[2]*Dpto. de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Camino de vera s/n, 46022 Valencia, Spain;*
e-mail: agiret@dsic.upv.es

Abstract

Many real life problems can be modeled as a scheduling problem. The main objective of these problems is to obtain optimal solutions in terms of processing time, cost and quality. Nowadays, energy-efficiency is also taken into consideration. However, these problems are NP-hard, so many search techniques are not able to obtain a solution in a reasonable time. In this paper, a genetic algorithm is developed to solve an extended version of the classical job-shop scheduling problem. In the extended version, each operation has to be executed by one machine and this machine can work at different speed rates. The machines consume different amounts of energy to process tasks at different rates. The evaluation section shows that a powerful commercial tools for solving scheduling problems was not able to solve large instances in a reasonable time, meanwhile our genetic algorithm was able to solve all instances with a good solution quality.

1 Introduction

Nowadays, many companies are not only facing complex and diverse economic trends of shorter product life cycles, quick changing science and technology, increasing customer demand diversity, and production activities globalization, but also enormous and heavy environmental challenges of global climate change (e.g. greenhouse effect) (Mestl *et al*., 2005), rapid exhaustion of various non-renewable resources (e.g. gas, oil, coal) (Yusoff, 2006), and decreasing biodiversity. All these requirements must be taken into account by the companies management systems (at all relevant levels: system, production and process) in order to become competitive in an ever increasing globalized market. The scheduling of manufacturing operations is a key technique that helps to achieve optimization of multi-objective criteria problems. In scheduling problems resources are assigned to tasks in order to minimize the completion time, reduce the use of resources, reduce idle time, etc. (Allahverdi *et al*., 2008; Bartak *et al*., 2010) Scheduling problems are widely discussed in the literature and two main approaches can be distinguished (Billaut *et al*., 2008):

- Classical deterministic methods, which consider that data are deterministic and that the machine environment is relatively simple. Some traditional constraints are taken into account (precedence constraints, release dates, due dates, preemption, etc.). The criterion to optimize is often standard (makespan). A number of methods have been proposed (exact methods, greedy algorithms, approximate methods, etc.), depending on the difficulty of the particular problem. These kind of studies are the most common in the specialized literature of scheduling problems.

- Online methods. Sometimes, the algorithm does not have access to all the data from the outset, the data become available step by step, or 'online'. Different models may be considered here, such as the complete list of tasks to schedule are not known, and the tasks appear one by one; or the duration of the tasks are not known in advance, among others.

In both cases, the job-shop scheduling problem (JSP) has been studied as an adequate modeling approach. It represents a particular case of scheduling problems where there are some specific resources or machines which have to be used to carry out some tasks. Many real life problems can be modeled as a JSP and can be applied in some variety of areas, such as production scheduling in the industry, departure and arrival times of logistic problems, the delivery times of orders in a company, etc. Most of the solving techniques try to find the optimality of the problem for minimizing the makespan, tardiness, flow time, etc.

Recently some works have focused on minimizing the energy consumption in scheduling problems (Dai *et al.*, 2013), mainly from the Operations Research Community (Mouzon *et al.*, 2007; Bruzzone *et al.*, 2012).

In JSP with voltage scaling, machines can consume different amount of energy to process tasks at different speeds (Malakooti *et al.*, 2013). By changing the voltage level, the frequency at which a processor executes a task is adjusted, and processing speed changes as a result. We focus our attention in a job-shop scheduling problem with different speed machine (JSMS). It represents an extension of the classical JSP ($J||C_{\max}$ according to classification scheme proposed in Blazewicz *et al.*, 1986), where each operation must be executed in a machine at a determined speed with a determined energy consumption (by a classical deterministic method). In this paper, a genetic algorithm (GA) is proposed to solve the JSP with speed scaling.

2 Problem description

Formally, the JSMS can be defined as follows. There exist a set of n jobs $\{J_1, \ldots, J_n\}$ and a set of m resources or machines $\{R_1, \ldots, R_m\}$.

Each job J_i consists of a sequence of T tasks $(\theta_{i1}, \ldots, \theta_{iT_i})$. Each task θ_{il} has a single machine requirement $R_{\theta_{il}}$ and a start time $st_{\theta_{il}}$ to be determined. The main difference with the traditional JSP is that each machine can work at different speeds. Thus, each task θ_{il} is associated to different durations $(p_{\theta_{il1}}, p_{\theta_{il2}}, \ldots, p_{\theta_{ilp}})$ and has different energy consumptions $e_{\theta_{il1}}, e_{\theta_{il2}}, \ldots, e_{\theta_{ilp}}$ depending on the machine in which it is executed.

In general, if the speed of a machine is high, the energy consumption increases, but the processing time of the task decreases, meanwhile if the speed is low, the energy consumption decreases and the processing time increases. For simplicity and without loss of generality, we consider three different energy consumptions and processing times for each task. The original processing time of each task is assigned to an energy consumption of the regular speed. If the processing time of a task is increased to 70%, the energy consumption is reduced to 20% (low speed). However, if the processing time of a task is reduced to 30%, the energy consumption is increased to 20% (high speed). Depending on the specific problem, this curve can vary, and therefore the proportion between processing time and energy consumption can significantly change.

A feasible schedule is a complete assignment of starting times to tasks that satisfies the following constraints: (i) the tasks of each job are sequentially scheduled, (ii) each machine can process at most one task at any time, (iii) no preemption is allowed. The objective is to find a feasible schedule that minimizes the completion time of all the tasks and the energy used.

3 Energy-efficiency

Statistical data in 2009 showed that the Germany industrial sector was responsible for ~47% of the total national electricity consumption, and the corresponding amount of CO_2 emissions generated by this consumption summed up to 18–20% (http://www.bmwi.de/EN/root.html). Thus, manufacturing companies are responsible of an important percentage of the environmental outcome and are forced to have manufacturing systems that demonstrate major potential to reduce environmental impacts (Duflou *et al.*, 2012).

Recently, there has been growing interest in the development of energy savings due to a sequence of serious environmental impacts and rising energy costs. Research on minimizing the energy consumption of manufacturing systems has focused on three perspectives: the machine level, the product level, and the

manufacturing system level. From the machine-level perspective, developing and designing more energy-efficient machines and equipment to reduce the power and energy demands of machine components is an important strategic target for manufacturing companies (Li *et al*., 2011; Neugebauer *et al*., 2011). Unfortunately, previous studies show that the share of energy demand for removal of metal material compared with the share of energy needed to support various functions of manufacturing systems is quite small (less than 30%) of total energy consumption (Dahmus & Gutowski, 2004). From the product-level perspective, modeling embodied product energy framework based on a product design viewpoint for energy reduction approach is beneficial to support the improvements of product design and operational decisions (Seow & Rahimifard, 2011; Weinert *et al*., 2011). It requires strong commercial simulation software to facilitate the analysis and evaluation of the embodied product energy. The results cannot be applied easily in most manufacturing companies, especially in small- and medium-sized enterprises due to the enormous financial investments required. From the manufacturing system-level perspective, thanks to decision models that support energy savings, it is feasible to achieve a significant reduction in energy consumption in manufacturing applications. In the specialized literature about production scheduling, the key production objectives for production decision models, such as cost, time and quality have been widely discussed. However, decreasing energy consumption in manufacturing systems through production scheduling has been rather limited.

One of the most well-known research works is the work of Mouzon *et al*. (2007), who developed several algorithms and a multiple objective mathematical programming model to investigate the problem of scheduling jobs on a single CNC machine in order to reduce energy consumption and total completion time. They pointed out that there was a significant amount of energy savings when non-bottleneck machines were turned off until needed; the relevant share of savings in total energy consumption could add up to 80%. They also reported that the inter-arrivals would be forecasted and therefore more energy-efficient dispatching rules could be adopted for scheduling. In further research, Mouzon and Yildirim (2008) proposed a greedy randomized adaptive search algorithm to solve a multi-objective optimization schedule that minimized the total energy consumption and the total tardiness on a machine. Fang *et al*. (2011) provided a new mixed-integer linear programming model for scheduling a classical flow shop that combined the peak total power consumption and the associated carbon footprint with the makespan. Yan and Li (2013) presented a multi-objective optimization approach based on weighted gray relational analysis and response surface methodology. Bruzzone *et al*. (2012) presented an energy-aware scheduling algorithm based on a mixed-integer programming formulation to realize energy savings for a given flexible flow shop that was required to keep fixed original job assignment and sequencing. Although the majority of the research on production scheduling has not considered energy-saving strategies completely, the efforts mentioned above provide a starting point for exploring an energy-aware schedule optimization from the viewpoint of energy consumption.

4 Modeling and solving a job-shop scheduling problem with different speed machine as a genetic algorithm

The most natural way to solve a traditional JSP is to represent all variables and constraints related to jobs, tasks and machines (Garrido *et al*., 2000; Huang & Liao, 2008) in order to be solved by a sound and completed search technique. The traditional objectives are to obtain solutions that minimize the typical objective functions presented in the literature (makespan, tardiness, completion time, etc.). It is well-known that this problem is NP-hard, so that optimal solutions can only be achieved for small instances. However, few techniques have been developed to minimize energy consumptions in these problems. Only in the last few years, some researchers have focused their attention to the machine level in order to solve the scheduling problem by minimizing the energy consumption (Dai *et al*., 2013). This requirement increases the complexity of the problem so it is not possible to obtain optimal solutions. This problem called JSMS must be solved by heuristic and metaheuristic techniques in order to obtain optimized solutions, mainly in large instances. To this end, in this paper we develop a GA to solve the JSMS. In Section 5, it can be observed that powerful commercial techniques were not able to solve large instances in a reasonable time; meanwhile small instances are solved by both techniques with similar solution quality.

The approach we propose uses a GA to solve the JSMS. GAs are adaptive methods which may be used to solve optimization problems (Beasley *et al*., 1993). They are based on the genetic process of biological

Algorithm 1: GeneticAlgorithm (JSMS, λ)

```
Begin
if (λ ≠ 0 and λ ≠ 1) then
  Initial-Population(Population, Size, Speed=Random(1,3));
else
  if (λ = 0) then
    Initial-Population(Population, Size, Speed=1);
  else
    Initial-Population(Population, Size, Speed=3);
  end if
end if
Evaluate-Fitness(Population);
while (Stopping criterion is not fulfilled) do
  Select-Parents(Population, Parent1, Parent2);
  Crossover(Parent1,Parent2,Offspring);
  mutation(Offspring,Offspring');
  Evaluate-Fitness(Offspring');
  Update-Population(Population,Offspring');
end while
Report Best Schedule;
End
```

organisms. Over many generations, natural populations evolve according to the principle of natural selection, that is survival of the fittest. At each generation, every new individual (chromosome) corresponds to a solution, which in our case is a schedule for the given JSMS instance. Before a GA can be run, a suitable encoding (or representation) of the problem must be devised. The essence of a GA is to encode a set of parameters (known as genes) and to join them together in order to form a string of values (chromosome). A fitness function is also required, which assigns a figure of merit to each encoded solution. The fitness of an individual depends on its chromosome and is evaluated by the fitness function. During the execution, parents must be selected for reproduction and recombined to generate offspring. Parents are randomly selected from the population, using a scheme which favors fitter individuals. Having selected two parents (Procedure Select—Parents in Algorithm 1, their chromosomes are combined, typically by using crossover and mutation mechanisms to generate better offspring that means better solutions). The process is iterated until a stopping criterion is satisfied.

Algorithm 1 shows the general steps of our GA. All functions are explained in detail in following subsections.

4.1 Chromosome encoding and decoding

In GA, a chromosome represents a solution in the search space. The first step in constructing the GA is to define an appropriate genetic representation (coding). A good representation is crucial because it significantly affects all the subsequent steps of the GA. Many representations for the JSP have been developed.

A chromosome is a permutation of the set of operations that represents a tentative ordering to schedule them, each one being represented by its job number. Figure 1 shows an example of a job-shop schedule with three jobs, where job 1 has two tasks, and both jobs 2 and 3 have three tasks. Each number in the chromosome cell (3, 2, 1, 3, 1, 2, 2, 3) represents the job of the task. The first number '3' represents the first task of the job 3. As it was shown in Varela *et al.* (2005), this encoding has a number of interesting properties for the classic JSP. For instance a random assignment of values 1, 2 and 3 generates a valid solution.

However, in the problem JSMS, the machine speed of each task has to be represented, therefore a new value must be added to each task in order to represent the machine speed. So a valid chromosome is $2n$ length, where n is the total number of tasks. Figure 2 shows the coding of a chromosome in JSMS. Such coding increases the former coding for adding the speed at which each task is processed. Thus, the first two digits 3 and 1 means that the first task of the job 3 is processed at speed 1. Again, a random assignment of values to tasks and speed generates a valid solution. When the chromosome representation is decoded each task starts as soon as possible following the precedence and machine constraints. With the machine speed

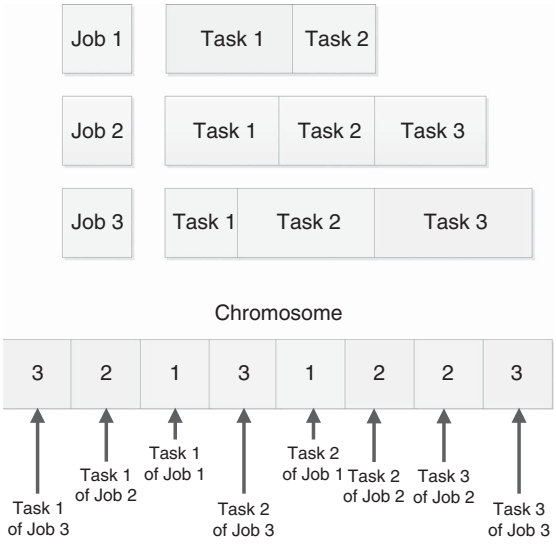

Figure 1 Codification of a chromosome in a job-shop scheduling problem

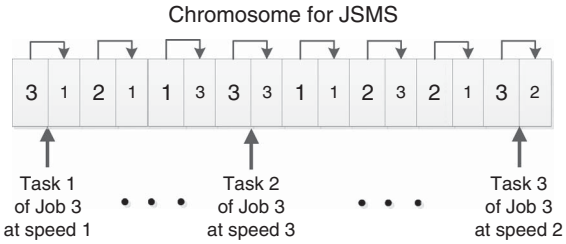

Figure 2 Codification of a chromosome in a job-shop scheduling problem with different speed machine (JSMS)

representation, the processing time of each task and energy consumption of the machine can be calculated, and therefore the makespan and total energy consumption.

4.2 Initial population and fitness

As we have pointed out before, each gene represents one task of the problem and the next gene represents the speed at which this task is processed. The position of each task determines its dispatch order, in this genome/solution. The initial chromosomes are obtained by a random permutation of tasks. The machine speed for each gene is also randomly generated among one of the possible speeds. Thus, the initial population is randomly generated to obtain feasible schedules (Procedure Initial—Population in Algorithm 1). According to the instance size, the population is properly tuned.

JSMS can be considered a multi-objective problem due to the fact that the goal is to minimize the makespan and also to minimize the energy consumption. However, both objectives are contrary so that minimize the makespan supposes to increase the speed of machines, and vice versa. Thus, in these problems no single optimal solution exists. Instead, a set of efficient solutions are identified to compose the Pareto front. Diverse techniques have been developed to solve multiple objective optimization problems. One of the most well-known methods for solving multiple objective optimization problems is the Normalized Weighted Additive Utility Function (NWAUF), where multiple objectives are normalized and added to form a utility function. NWAUF has been used widely for multiple objective optimization problems due to its simplicity and natural ability to identify efficient solutions. Let f_{ij} be the ith objective function value of alternative j. Then, the NWAUF for alternative j with k objectives is defined as

$$U_j = w_1 f'_{1j} + w_2 f'_{2j} + \ldots + w_k f'_{kj} \tag{1}$$

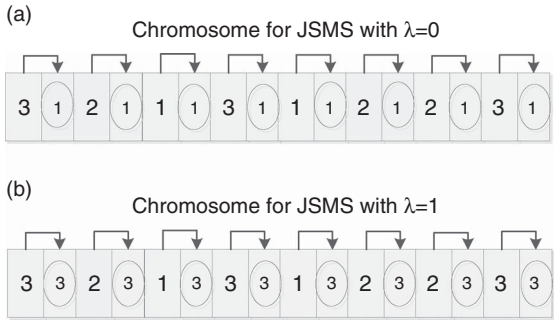

Figure 3 Initial population for $\lambda = 0$ and $\lambda = 1$. JSMS = job-shop scheduling problem with different speed machine

where $w_1, w_2, \ldots, w_k$ are weights of importance and $f'_{1j}, f'_{2j}, \ldots, f'_{kj}$ are normalized values of $f_{1j}, f_{2j}, \ldots, f_{kj}$. By normalizing different objectives, all objectives are evaluated in the same scale. Weights show decision maker's preference for each objective where, $\sum_{i=1}^{k} w_i = 1$ and $0 \leqslant w_i \leqslant 1$ for $i = 1, \ldots, k$. Using this utility function, the multiple objective optimizations can now be solved as a single objective optimization problem.

The definition of fitness function is just the reciprocal of the objective function value. The objective is to find a solution that minimizes the multi-objective makespan and energy consumption. Following NWAUF rules, our fitness function $F(i)$ (2) is a convex combination between the normalized values of makespan and energy consumption of solution i.

$$F(i) = \lambda \times NormMakespan\,(i) + (1-\lambda) \times NormEnergy\,(i) \tag{2}$$

$$NormMakespan\,(i) = \frac{Makespan\,(i)}{MaxMakespan} \tag{3}$$

$$NormEnergy\,(i) = \frac{SumEnergy\,(i)}{MaxEnergy} \tag{4}$$

where $\lambda \in [0, 1]$. *NormMakespan* (3) is the makespan divided by the maximum makespan value in a GA execution when the λ value is equal to 0 (*MaxMakespan*). *MaxMakespan* values can be found in the benchmark section of our Web page[1]. *NormEnergy* (4) is calculated by summing the energy used in the execution of all the tasks, and then dividing the result by the maximum energy (*MaxEnergy*). *MaxEnergy* is the sum of the energy needed to execute all tasks at top speed.

Once the λ parameter is set for the fitness function (2), the initial population can be generated in a specific way. Thus, for $\lambda = 0$, the objective function is only focused to reduce the energy consumption ($F = NormEnergy$), so the initial population can be randomly generated to order the tasks, but the corresponding speeds are fixed to the lowest value (see Figure 3a). In the same way, if $\lambda = 1$, the objective function is only focused to reduce makespan ($F = NormMakespan$), so the initial population can also be randomly generated to order the tasks, but the corresponding speeds are fixed to the highest value (see Figure 3b). For $\lambda \in]0, 1[$, the speed of each task can be appropriately generated.

4.3 Crossover operator

For chromosome mating, our GA uses a (job, energy)-based order crossover. Thus, given two parents, a set of pairs (job, energy) of a random job is selected from the first parent and copied in the same position to the offspring. Afterwards, the set of pairs (job, energy) of the remaining jobs are translated from the second parent to the offspring in the same order (Procedure Crossover in Algorithm 1).

[1] `http://gps.webs.upv.es/jobshop/`

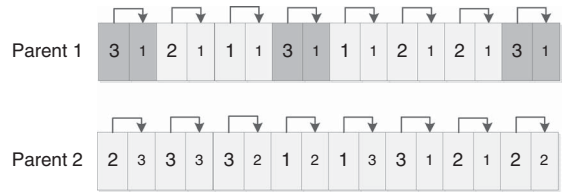

Figure 4 Two parents for the crossover operator

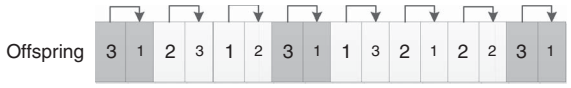

Figure 5 Offspring for the crossover operator

We clarify how this technique works in the next example. Let us consider the following two parents (see Figure 4).

If the selected subset of jobs from the first parent just includes the job 3 (dark genes in Figure 4), the generated offspring is showed in Figure 5.

Hence, this technique maintains for a machine a subsequence of operations in the same order as they are in parent 1 and the remaining ones in the same order as they appear in parent 2. The crossover is applied in a dual way, so two parents generate two offspring (parent 1–parent 2) and (parent 2–parent 1). Parent couples are selected shuffling population and choosing each couple two by two, so all individual will be selected but only some couples will be crossed in accordance to crossover probability.

4.4 Mutation operator

The two offsprings generated with crossover operation can be also mutated in accordance to the mutation probability (Procedure Mutation in Algorithm 1). Two pair (task, energy) position of chromosome child are randomly chosen (position ‘a’ and position ‘b’), where ‘a’ must be lower than ‘b’. Pairs between ‘a’ and ‘b’ are shuffled randomly, also in each gene machine speed values are randomly changed. In this step, the speeds of the machines in tasks are also randomly modified.

Finally, tournament replacement among every couple of parents and their offspring is done to obtain the next generation (Procedure Update—Population in Algorithm 1).

5 Evaluation

In this section, we evaluate the behavior of our GA against a successful and well-known commercial solver IBM ILOG CPLEX CP Optimizer tool 12.5 (CP Optimizer). It is a commercial solver embedding powerful constraint propagation techniques and a self-adapting large neighborhood search method dedicated to scheduling (Laborie, 2009). This solver is expected to be very efficient for a variety of scheduling problems as it is pointed out in IBM (2007), in particular when the cumulative demand for resources exceeds their availability.

These algorithms have been evaluated with extended benchmarks of the typical JSP. The extension has been focused on assigning different speeds and durations to each task (as we pointed out in Section 3). The extension with machines working at different speeds has been implemented considering that each task is executed by a machine and it has different optional modes where each represents the duration of the task and an associated energy consumption (Salido *et al*., 2013).

To this end, we extend the benchmarks proposed in Agnetis *et al*. (2011) and Watson *et al*. (1999) because to the best of our knowledge there are no benchmarks for JSPs that incorporate different speeds and energy consumptions. All instances are characterized by the number of jobs (j), the number of machines (m), the maximum number of tasks by job ($v_{\max}$) and the range of processing times (p).

Table 1 Results of small Agnetis instances

	3_5_10						7_10_100					
	CP Optimizer			Genetic			CP Optimizer			Genetic		
λ	Mk	En	F	Mk	En	F	Mk	En	F	Mk	En	F
0	71.4	84.4	0.553762	65.8	84.4	0.553762	1088.4	1571.4	0.533616	1006.3	1571.4	0.533616
0.1	65.2	84.5	0.556581	65.2	84.5	0.556581	999.3	1572.6	0.540932	999.3	1572.6	0.540931
0.2	64.4	84.7	0.558703	64.4	84.7	0.558703	987.2	1576.5	0.547508	987.2	1576.5	0.547509
0.3	63.2	85.2	0.559422	63.2	85.2	0.559422	922.2	1613.3	0.550868	926.9	1610	0.551018
0.4	597	88.1	0.557816	59.7	88.1	0.557816	885.9	1649.2	0.550650	891	1642.9	0.550638
0.5	53.9	94.3	0.547187	54.2	93.9	0.547270	838.8	1716	0.545249	847.5	1704.8	0.545938
0.6	48.4	104.2	0.529935	48.9	103.2	0.529687	779.1	1859.3	0.535095	782.4	1845.7	0.535361
0.7	45.3	111.9	0.500419	45	112.7	0.500130	708.5	2068.4	0.511331	703.9	2099.5	0.512783
0.8	42.2	123.4	0.461368	42.2	123.5	0.461509	651.8	2346	0.475184	642.4	2418.9	0.475810
0.9	41	133.2	0.414361	41	133.7	0.414712	626	2560.7	0.428228	626	2573.3	0.428603
1	41	143.1	0.363050	41	145.3	0.363050	625.9	2664.1	0.378956	625.9	2773.4	0.378956

Mk = makespan; En = energy consumption; F = fitness function.

In Agnetis instances, j is set to 3 and these can be represented as $m\_v_{\max}\_p$, and the number of operators was not considered in this study, so we fixed it to the number of machines. The population size was set to 200 individual for Agnetis instances and 400 for Watson instances. These values were selected by testing different alternatives and selecting the best results.

We consider two types of instances: small and large Agnetis instances:

- $j = 3$; $m = 3, 5, 7$; $v_{\max} = 5, 7, 10$; $p = [1, 10], [1, 50], [1, 100]$
- $j = 3$; $m = 3$; $v_{\max} = 20, 25, 30$; $p = [1, 50], [1, 100], [1, 200]$

In Watson instances follow the same characterized, but in this case the variable that changes is the number of jobs (j):

- $j = 50, 100, 200$; $m = 20$; $v_{\max} = 20$; $p = [1, 100]$

For each type of instance we work with 10 instances so the results presented in this section are always the averages value. We have modeled the instances to be solved by the CP Optimizer. Moreover, we have extended the original instances as explained in Escamilla *et al*. (2014). These instances can be found in the Web page[2].

5.1 Comparative study between CP Optimizer and genetic algorithm

CP and GA techniques try to minimize the multi-objective makespan and energy consumption. The weight of each objective can be changed by λ parameter, following the expression (2). In order to compare both techniques, the validation tests have been executed in an Intel Core2 Quad CPU Q9550, 2.83 GHz and 4 GB RAM computer with Ubuntu 12.04 operating system. The small Agnetis instances were executed during 5 seconds and the large Agnetis and Watson instances had a 100 seconds time-out. The next tables present the most important parameter to be analyzed: $\lambda \in [0, 1]$ that represents the weight given to makespan and energy consumption, Mk is the makespan, En is the energy consumption, and F is the fitness function. The objective is to obtain the lowest value of F.

Table 1 shows the results for two small Agnetis instances, the smallest (3_5_10) and the largest (7_10_100) of this group. The results for the instances 3_5_10 show that the F value was equal or almost equal in both CP Optimizer and GA. Furthermore, there were small differences in all λ values for instance

[2] `http://gps.webs.upv.es/jobshop/`

Table 2 Results of large Agnetis instances

	3_25_100					
	CP Optimizer			Genetic		
λ	Mk	En	F	Mk	En	F
0	3160	3827.1	0.533532	3096	3829	0.533805
0.1	2768.1	3827.6	0.537461	2781.7	3827.9	0.537786
0.2	2719.3	3842.5	0.540966	2764.8	3845.5	0.543204
0.3	2597.9	3904.6	0.542188	2657.3	3920.2	0.547324
0.4	2480.7	4005.8	0.540172	2495.3	4068.7	0.546693
0.5	2342	4181.6	0.533724	2317.1	4257.3	0.536410
0.6	2147	4548.6	0.520427	2118.3	4617.4	0.520423
0.7	1935.5	5075.6	0.492575	1943.7	5097.4	0.494838
0.8	1806.2	5666	0.456913	1791.4	5726.2	0.456512
0.9	1725.9	6251.3	0.408634	1732.2	6311	0.410335
1	1673.4	6732.2	0.346046	1711.8	6797.2	0.353841

Mk = makespan; En = energy consumption; F = fitness function.

Table 3 Results of Watson instances

	CP Optimizer			Genetic		
λ	F_50	F_100	F_200	F_50	F_100	F_200
0	0.53408	0.53061	0.77288	0.610346	0.63776	0.68297
0.1	0.55899	0.56114	No sol.	0.63994	0.66249	0.69391
0.2	0.58329	0.59138	No sol.	0.65530	0.68276	0.71923
0.3	0.60836	0.62144	No sol.	0.66789	0.69951	0.73313
0.4	0.63038	0.65117	No sol.	0.67253	0.70370	0.73440
0.5	0.65334	0.68317	No sol.	0.66715	0.69871	0.73288
0.6	0.66936	0.69793	No sol.	0.65396	0.69041	0.72827
0.7	0.65937	0.69234	No sol.	0.63458	0.67832	0.72394
0.8	0.64346	0.68051	No sol.	0.61773	0.66570	0.71953
0.9	0.63197	0.66429	No sol.	0.59558	0.65161	0.7136
1	0.52675	0.63500	0.69452	0.57309	0.64023	0.70732

7_10_100. These results show that both algorithms maintained the same behavior for small instances (the difference is in the fourth decimal).

In large Agnetis instances, the results were also similar for all the instances. Table 2 shows the results for the instances 3_25_100 as an example. The difference of F value between CP Optimizer and GA was almost in the third decimal in most cases. It must be taken into account that for $\lambda = 0.6$ or $\lambda = 0.8$, the F value of our GA was lower than in CP Optimizer. This is due to the fact that the initial population, for $\lambda = 1$ is composed of high speed values (3).

Table 3 shows the F values for Watson instances. In Agnetis instances, the maximum number of operations is 90 in instances 3_30_p. However, in Watson instances, the number of operations is ranged between 1000 ($j = 50$ and $v_{\max} = 20$) and 4000 operations ($j = 200$ and $v_{\max} = 20$). Therefore, Watson instances are much larger than Agnetis instances. It can be observed that both algorithms were able to solve all instances with 50 and 100 jobs. The results for these instances were better for CP Optimizer for λ values <0.6, meanwhile our GA had better results for $\lambda \in [0.6, 0.9[$. Figure 6 shows the average F value of 50 and 100 jobs for Watson instances. It can be observed that although both algorithms have similar behavior, GA

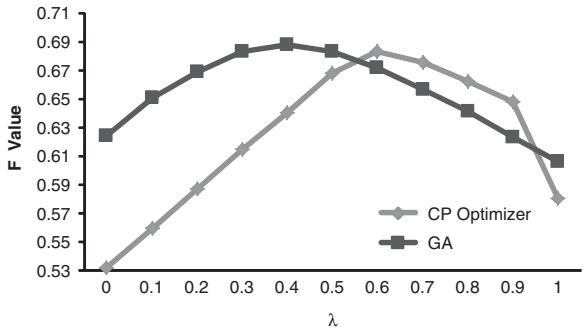

Figure 6 Average F value (F_50 and F_100) for Watson instances

is mainly focused on minimizing makespan (highest value for $\lambda = 0.4$) meanwhile CP Optimizer is mainly focused on minimizing energy consumption (highest value for $\lambda = 0.6$).

However, for instances of 200 jobs, CP Optimizer was unable to solve almost all instances ranged for $\lambda \in]0, 1[$. This means CP Optimizer is not able to solve large-scale instances in a reasonable time so metaheuristic techniques are needed to obtain optimized solutions in a given time.

6 Conclusions and further works

Many real-world problems can be modeled as a scheduling problem. In this case we are working with an extended version of the classical JSP in which machines can consume different amounts of energy to process tasks at different rates. Each operation has to be executed by one machine and this machine has the possibility to work at different speeds. In this work, we develop a GA to solve the JSMS problem. The representation of energy consumption in the chromosome gives the opportunity to guide the search toward an optimized solution in an efficient way. A comparative study was carried out to analyze the behavior of our GA against a well-known solver: IBM ILOG CPLEX CP Optimizer. The evaluation shows that our GA had a similar behavior than CP Optimizer for small instances. However, for large instances, CP Optimizer was unable to solve them in the given time, meanwhile our GA could solve all instances with the same optimality degree. Thus, our technique can be useful for application in large-scale scheduling problems.

In further works, we aim to add more techniques to GA algorithm such as local search techniques to improve the obtained solutions. It can also be taken into consideration adding a post-process by increasing the speed of the machine that executes tasks in the critical path. Furthermore, we will take into account robustness and energy aware due to the fact that energy-aware solutions can be considered more robust than makespan-optimized solutions (Salido *et al.*, 2013).

Acknowledgments

This research has been supported by the Spanish Government under research project MINECO TIN2013-46511-C2-1.

References

Agnetis, A., Flamini, M., Nicosia, G. & Pacifici, A. (2011). A job-shop problem with one additional resource type. *Journal of Scheduling* **14**(3), 225–237.

Allahverdi, A., Ng, C.T., Cheng, T.C.E. & Kovalyov, M. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research* **187**(3), 985–1032.

Bartak, R., Salido, M. & Rossi, F. (2010). New trends in constraint satisfaction, planning, and scheduling: a survey. *The Knowledge Engineering Review* **25**(3), 249–279.

Beasley, D., Martin, R. & Bull, D. (1993). An overview of genetic algorithms: part 1. Fundamentals. *University Computing* **15**, 58–58.

Billaut, J., Moukrim, A. & Sanlaville, E. (2008). *Flexibility and Robustness in Scheduling*. Wiley.

Blazewicz, J., Cellary, W., Slowinski, R. & Weglarz, J. (1986). Scheduling under resource constraints-deterministic models. *Annals of Operations Research* **7**, 1–356.

Bruzzone, A., Anghinolfi, D., Paolucci, M. & 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.

Dahmus, J. & Gutowski, T. 2004. An environmental analysis of machining. In *ASME International Mechanical Engineering Congress and RD&D Exposition*.

Dai, M., Tang, D., Giret, A., Salido, M.A. & Li, W. (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.

Duflou, J., Sutherland, J., Dornfeld, D., Herrmann, C., Jeswiet, J., Kara, S., Hauschild, M. & Kellens, K. (2012). Towards energy and resource efficient manufacturing: a processes and systems approach. *CIRP Annals-Manufacturing Technology* **61**(2), 587–609.

Escamilla, J., Salido, M. A., Giret, A. & Barber, F. (2014). A metaheuristic technique for energy-efficiency in job-shop scheduling. In *COPLAS' 2014: ICAPS Workshop on Constraint Satisfaction Techniques for Planning and Scheduling*, 42–50.

Fang, K., Uhan, N., Zhao, F. & Sutherland, J. (2011). A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. *Journal of Manufacturing Systems* **30**(4), 234–240.

Garrido, A., Salido, M. A., Barber, F. & Lopez, M. (2000). Heuristic methods for solving job-shop scheduling problems. In *ECAI-2000 Workshop on New Results in Planning, Scheduling and Design*, 36–43.

Huang, K. & Liao, C. 2008. Ant colony optimization combined with taboo search for the job shop scheduling problem. *Computers & Operations Research* **35**(4), 1030–1046.

IBM 2007. Modeling with IBM ILOG CP optimizer – practical scheduling examples. IBM.

Laborie, P. 2009. IBM ILOG CP optimizer for detailed scheduling illustrated on three problems. In *Proceedings of the 6th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR09)*, 148–162.

Li, W., Zein, A., Kara, S. & Herrmann, C. (2011). An investigation into fixed energy consumption of machine tools. In *Glocalized Solutions for Sustainability in Manufacturing*, Hesselbach, J. & Herrmann, C. (eds), 268–273. Springer.

Malakooti, B., Sheikh, S., Al-Najjar, C. & Kim, H. (2013). Multi-objective energy aware multiprocessor scheduling using bat intelligence. *Journal of Intelligent Manufacturing* **24**(4), 805–819.

Mestl, H. E., Aunan, K., Fang, J., Seip, H. M., Skjelvik, J. M. & Vennemo, H. (2005). Cleaner production as climate investment integrated assessment in Taiyuan City, China. *Journal of Cleaner Production* **13**(1), 57–70.

Mouzon, G. & Yildirim, M. 2008. A framework to minimise total energy consumption and total tardiness on a single machine. *International Journal of Sustainable Engineering* **1**(2), 105–116.

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

Neugebauer, R., Wabner, M., Rentzsch, H. & Ihlenfeldt, S. (2011). Structure principles of energy efficient machine tools. *CIRP Journal of Manufacturing Science and Technology* **4**(2), 136–147.

Salido, M. A., Escamilla, J., Barber, F., Giret, A., Tang, D. & Dai, M. (2013). Energy-aware parameters in job-shop scheduling problems. In *GREEN-COPLAS 2013: IJCAI 2013 Workshop on Constraint Reasoning, Planning and Scheduling Problems for a Sustainable Future*, 44–53.

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

Varela, R., Serrano, D. & Sierra, M. (2005). New codification schemas for scheduling with genetic algorithms. In Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach, 11–20. Springer.

Watson, J.-P., Barbulescu, L., Howe, A. E. & Whitley, L. D. (1999). Algorithm performance and problem structure for flow-shop scheduling. In AAAI/IAAI, 688–695.

Weinert, N., Chiotellis, S. & Seliger, G. (2011). Methodology for planning and operating energy-efficient production systems. *CIRP Annals-Manufacturing Technology* **60**(1), 41–44.

Yan, J. & Li, L. 2013. Multi-objective optimization of milling parameters the trade-offs between energy, production rate and cutting quality. *Journal of Cleaner Production* **52**, 462–471.

Yusoff, S. 2006. Renewable energy from palm oil—innovation on effective utilization of waste. *Journal of Cleaner Production* **14**(1), 87–93.