Robust Schedules for Tardiness Optimisation in Job Shop with Interval Uncertainty

Hernán Díaz
Dep. of Computing, University of Oviedo, Spain
diazhernan@uniovi.es

Juan José Palacios
Dep. of Computing, University of Oviedo, Spain
palaciosjuan@uniovi.es

Irene Díaz
Dep. of Computing, University of Oviedo, Spain
sirene@uniovi.es

Camino R. Vela
Dep. of Computing, University of Oviedo, Spain
crvela@uniovi.es

Inés González-Rodríguez
Dep. of Maths, Stats and Computing, University of Cantabria, Spain
gonzalezri@unican.es

September 22, 2021

Abstract

This paper addresses a variant of the job shop scheduling problem with total tardiness minimisation where task durations and due dates are uncertain. This uncertainty is modeled with intervals. Different ranking methods for intervals are considered and embedded into a genetic algorithm. A new robustness measure is proposed to compare the different ranking methods and assess their capacity to predict "expected delays" of jobs. Experimental results show that dealing with uncertainty during the optimisation process yields more robust solutions. A sensitivity analysis also shows that the robustness of the solutions given by the solving method increases when the uncertainty grows.

Keywords: Job shop scheduling; Total tardiness; Interval uncertainty; Robustness

1 Introduction

Scheduling problems consist in allocating a set of resources to perform a set of tasks under a set of given constraints while optimising a certain performance metric. Solving these problems has lead to reducing energy and/or material-handling costs, as well as time, and overall improving the efficiency of chain production [27, 34]. The job shop, or JSP in short, has been considered one of the most relevant scheduling problems, being a good model for many practical applications as well as a challenge to the research community due to its complexity. The latter is the reason why metaheuristic search techniques are especially suited for solving the JSP [40].

The existing literature on job shop scheduling problems focuses mainly on minimising the execution timespan of the project (known as makespan). However, on-time fulfilment has become especially relevant in modern pull-oriented supply chain systems concerned with meeting customers' demand in terms of due dates; a tardy job may result in delay-compensation cost, customer dissatisfaction or loss of reputation among others. In the case of complex manufacturing systems, tardy jobs in one of the stages can cause serious disruptions and delays in subsequent stages, with the associated costs. For these reasons, due-date related criteria have been gaining importance in recent years [28, 30].

Traditionally in scheduling, it has been assumed that design variables such as task processing times or due dates are deterministic. However, in real-world production scheduling problems such variables are quite often characterised vaguely due to the available information being incomplete or imprecise. In fact, the necessity of dealing with imprecision in real world problems has been a long-term research challenge. The most common approach to handling uncertainty in scheduling is that of stochastic scheduling, modelling the duration of tasks by probability distributions [34]. However, even if the uncertain parameters are independent random variables, finding the distribution of the objective function is in general intractable. Also, as pointed out in [12], probability distributions permit to model variability of repetitive tasks, but not uncertainty due to lack of information, and they may not be adequate when different job shop projects take place under slightly different conditions. Fuzzy scheduling takes an alternative approach, modelling uncertain variables as fuzzy numbers or fuzzy intervals, that is, possibility distributions representing more or less plausible values [13]. This approach is computationally more appealing and it presupposes less knowledge and it has been extensively used to model uncertainty in scheduling problems, both for durations [5] and due dates [10]. However, the simplest way of representing uncertainty are intervals: assigning a time interval I to an activity duration (respectively, a due date) means that the actual duration of this activity (respectively due date) will take some value in that interval, but it is not possible at present to predict exactly which one it will be [15]. Interval uncertainty is present as soon as information is incomplete and it does not assume any further knowledge. Intervals naturally arise if an expert is reluctant or unable to provide a particular value for task durations and feels that estimating a minimal and a maximal duration is more realistic. Also, the completion of a job may depend on several uncertain events such as changing customer orders or dependencies on other components of a larger manufacturing system, so giving a range of possible due date values seems better suited than risking a crisp and unrealistic due date. Under such circumstances, interval scheduling gives the possibility of focussing on significant scheduling decisions and producing robust solutions. Also, it represents a first step towards solving problems in the other uncertain frameworks. Indeed, an interval can be seen as a uniform probability distribution or the support of an unknown probability distribution, as noted in [2]. An interval is also a particular case of a fuzzy interval where all values in its support are equally possible and, in general, the α-cuts of fuzzy intervals are intervals [13].

Interval uncertainty is not new in scheduling, although contributions in the literature are still scarce. In [24], a genetic algorithm is proposed for a JSP with both processing times and due dates represented by intervals that minimises the total tardiness with respect to job due dates. A population-based neighborhood search for a interval job shop, but with the objective of the makespan is presented in [23]. A multiobjective interval JSP with non-resumable jobs and flexible maintenance is solved in [25] by means of a multiobjective ABC algorithm that minimises both makespan and total tardiness. In [26], a hybrid between PSO and a genetic algorithm is used to solve a flexible JSP with interval processing times as part of a larger integrated planning and scheduling problem.

Metaheuristic search methods are especially suited for scheduling problems due to their complexity. In particular, genetic algorithms, on their own or combined with other metaheuristic methods, have proved to be successful in tackling the job shop, which is a NP-hard problem [16]. For instance, among the state-of-the-art methods for the deterministic job shop with total weighted tardiness, we find a genetic algorithm with an iterated local search [14] and a hybrid genetic algorithm with tabu search [17]. In [39], a combination of a genetic algorithm with simulated annealing is used to minimise the makespan. For flexible job shop problems, genetic algorithms have gained great popularity [3], and they are used as well in [35] for open shop scheduling problems. When uncertainty is present, a genetic algorithm with heuristic seeding for the fuzzy flexible job shop problem is used in [31], and a genetic algorithm hybridised with tabu search is used in [37] for the fuzzy job shop maximising due-date satisfaction.

In optimisation problems, and in scheduling in particular, being able to compare solutions in terms of the objective function values is crucial. Under interval uncertainty, objective function values may also be intervals that need to be compared. However, there is no natural ordering in the set of intervals. Instead, different ranking methods can be found in the literature (cf. [8, 21]). In consequence, one of these ranking methods need to be adopted during the optimisation process in order to compare solutions. In the above-mentioned works, the authors consider a single interval ranking to compare and select solutions, without analysing whether or how does the choice of a ranking affects the quality of the obtained schedule. However, as shown in [33] for the fuzzy job shop problem, the choice of the ranking method can have a big impact on the obtained solution and its robustness in real scenarios, and therefore, it should be analysed.

The high-level idea of solution robustness can be translated into many different robustness measures, depending on the source of uncertainties, the nature of the objective function and the context of the problem [4, 36]. The most traditional approach based on min-max or min-max regret criteria consists in finding solutions with the best possible performance in the worst case and is motivated by practical applications where an anticipation of the worst case is crucial [1]. There are however other cases where the worst case is not critical and, instead, solutions with an overall acceptable performance are preferred [20]. This is the stance taken in the definition of a robustness measure for tardiness minimisation in job shop with interval uncertainty given in [11]. To our knowledge, this is the only existing proposal in the literature for this variant of the problem but, as we shall argue in the sequel, it presents some flaws when tardiness minimisation is considered.

In the following, we consider the job shop scheduling problem with intervals modelling both uncertain durations and uncertain due dates. The objective is to minimise the total tardiness, an interval in this case. We propose several variants of a genetic algorithm with different interval ranking methods to study their influence on the optimisation process. We also consider modelling the processing times and due dates with a scalar value to assess the actual benefits of incorporating the uncertainty into the search. We also propose a new robustness measure to compare the schedules' performance in terms of unexpected delays. This measure is also used in a sensitivity analysis to test the behaviour of the proposed algorithms when faced with increasing uncertainty.

This paper is an extended version of the work presented at the 15th International Conference on Hybrid Artificial Intelligence Systems (HAIS'20) and published in [11]. The contributions of this new extended version are the following:

- A new robustness measure, better suited for the total tardiness objective function, is introduced in Section 3.

- As a consequence of the change in the robustness measure, all the experiments related to this metric have been rerun and are presented in Section 5.
- It was mentioned in the conclusions of [11] that there was a need to thoroughly analyse the influence of different ranking methods in order to make a proper choice. We address this issue here by performing a separate parameter tuning for each variant of the GA depending on the ranking method used. The experimental results in 5 are then given for the individually optimised variants of the GA. Based on these updated results, we incorporate a more detailed comparison of all ranking methods and using scoring ranking rules to choose the most adequate in terms of robustness.
- We include in Section 5 a sensitivity analysis to study the behaviour of the GA variants when the uncertainty in the problem increases.

2 Problem Definition

The classical *job shop scheduling problem*, consists of a set of jobs $J = \{J_1, \dots, J_n\}$ to be scheduled on a set of physical resources or machines $M = \{M_1, \dots, M_m\}$, subject to a set of constraints. There are *precedence constraints*, so each job J_j, consists of $m_j \leq m$ tasks $(o(j,1), \dots, o(j,m_j))$ to be sequentially scheduled. There are also *capacity constraints*, whereby each task $o(j,l)$ requires the uninterrupted and exclusive use of a specific machine $\nu_{o(j,l)} \in M$ for its whole processing time $p_{o(j,l)}$. Additionally, each job J_j has a desirable completion due date d_j.

A solution to this problem is a *schedule* $\boldsymbol{s}$, i.e. an allocation of starting times $s_{o(j,l)}$ for each task $o(j,l)$ that satisfies all constraints and is optimal according to some criterion, in our case, minimum total tardiness with respect to due dates.

A schedule $\boldsymbol{s}$ establishes an order π among the tasks requiring the same machine. Conversely, given a task processing order π, the schedule $\boldsymbol{s}(\pi)$ can be computed as follows. Let $s_{o(j,l)}(\pi)$ denote the starting time of task $o(j,l)$ given π, $c_{o(j,l)}(\pi) = s_{o(j,l)}(\pi) + p_{o(j,l)}$ its completion time, and $PM_{o(j,l)}(\pi)$ its predecessor in its required machine. The starting time of a task $o(j,l)$ is given by $s_{o(j,l)}(\pi) = \max(c_{o(j,l-1)}(\pi), c_{PM_{o(j,l)}(\pi)}(\pi))$. If task $o(j,l)$ has no machine predecessor in π, then $c_{PM_{o(j,l)}(\pi)}(\pi) = 0$ (similarly, $c_{o(j,0)} = 0$). The total tardiness of the schedule is given by $T_{tot}(\pi) = \sum_{j=1}^{n} T_j(\pi)$, where $T_j(\pi)$ is the tardiness of job J_j according to π, $T_j(\pi) = \max(0, c_{o(j,m_j)}(\pi) - d_j)$.

2.1 Uncertain Processing Times and Due Dates

As argued in Section 1, in real life applications is common to encounter uncertainty regarding the exact time it will take to process a task on the final due dat for a job. If only an upper and a lower bound of each duration and due date are known, uncertainty can be represented as a closed interval $\mathbf{a} = [\underline{a}, \overline{a}] = \{x \in \mathbb{R} : \underline{a} \leq x \leq \overline{a}\}$, with $\underline{a}$ and $\overline{a}$ the corresponding lower and upper bounds.

Let $\mathbb{IR}$ denote the set of closed intervals. The job shop problem with total tardiness minimisation requires three arithmetic operations on $\mathbb{IR}$: addition, subtraction and maximum. These are defined by extending the corresponding operations on real numbers [29], so given two intervals $\mathbf{a} = [\underline{a}, \overline{a}], \mathbf{b} = [\underline{b}, \overline{b}] \in \mathbb{IR}$, the addition is expressed as $[\underline{a} + \underline{b}, \overline{a} + \overline{b}]$, the subtraction as $[\underline{a} - \overline{b}, \overline{a} - \underline{b}]$ and the maximum as $[\max(\underline{a}, \underline{b}), \max(\overline{a}, \overline{b})]$.

Comparisons are a key point in scheduling, as the “best” schedule should be the one with “minimal” total tardiness (an interval). However, there is no natural total order in the set of intervals, so an interval ranking method needs to be considered among those proposed in the literature [21]. However, no consensus exists in the literature as to which ranking is the most appropriate for scheduling problems. Furthermore, it has been shown in a different uncertainty framework [33] that the choice of ranking method to compare uncertain objective function values noticeable affects the robustness of the obtained solution. Among the multiple existing rankings in $\mathbb{IR}$, here we consider the following:

$$\mathbf{a} \leq_{Lex1} \mathbf{b} \Leftrightarrow \underline{a} < \underline{b} \vee (\underline{a} = \underline{b} \wedge \overline{a} \leq \overline{b}), \tag{1}$$

$$\mathbf{a} \leq_{Lex2} \mathbf{b} \Leftrightarrow \overline{a} < \overline{b} \vee (\overline{a} = \overline{b} \wedge \underline{a} \leq \underline{b}), \tag{2}$$

$$\mathbf{a} \leq_{YX} \mathbf{b} \Leftrightarrow \underline{a} + \overline{a} < \underline{b} + \overline{b} \vee (\underline{a} + \overline{a} = \underline{b} + \overline{b} \wedge \overline{a} - \underline{a} \leq \overline{b} - \underline{b}), \tag{3}$$

$$\mathbf{a} \leq_{MP} \mathbf{b} \Leftrightarrow m(\mathbf{a}) \leq m(\mathbf{b}) \text{ with } m(\mathbf{a}) = (\underline{a} + \overline{a})/2. \tag{4}$$

It can be seen that (1), (2) and (3) actually define total order relations in $\mathbb{IR}$ [8]. Both (1) and (2) are derived from a lexicographical order of interval extreme points. The ranking in expression (3) is proposed in [38], and the last one (midpoint order) is a particular case of the classical Hurwitz criterion and is equivalent to the one used in [24] for interval JSP.

2.2 The JSP with Interval Uncertainty

Given the above, the *Interval Job Shop Scheduling Problem* or *IJSP* for total tardiness minimisation can be formulated as follows:

$$\min_R \mathbf{T_{tot}} = \sum_{j=1}^{n} \mathbf{T}_j \tag{5}$$

$$\text{s.t.:}\quad \underline{T}_j = \max(0, \underline{c}_{o(j,m_j)} - \overline{d_j}) \tag{6}$$

$$\overline{T}_j = \max(0, \overline{c}_{o(j,m_j)} - \underline{d_j}) \tag{7}$$

$$\underline{c}_{o(j,l)} = \underline{s}_{o(j,l)} + \underline{p}_{o(j,l)} \qquad 1 \le l \le m_j, 1 \le j \le n \tag{8}$$

$$\overline{c}_{o(j,l)} = \overline{s}_{o(j,l)} + \overline{p}_{o(j,l)} \qquad 1 \le l \le m_j, 1 \le j \le n \tag{9}$$

$$\underline{s}_{o(j,l)} \ge \underline{c}_{o(j,l-1)} \qquad 1 \le l \le m_j, 1 \le j \le n \tag{10}$$

$$\overline{s}_{o(j,l)} \ge \overline{c}_{o(j,l-1)} \qquad 1 \le l \le m_j, 1 \le j \le n \tag{11}$$

$$\underline{s}_{o(j,l)} \ge \underline{c}_{o(j',l')} \ \vee\ \underline{s}_{o(j',l')} \ge \underline{c}_{o(j,l)} \qquad \forall o(j,l) \ne o(j',l') : \nu_{o(j,l)} = \nu_{o(j',l')} \tag{12}$$

$$\overline{s}_{o(j,l)} \ge \overline{c}_{o(j',l')} \ \vee\ \overline{s}_{o(j',l')} \ge \overline{c}_{o(j,l)} \qquad \forall o(j,l) \ne o(j',l') : \nu_{o(j,l)} = \nu_{o(j',l')} \tag{13}$$

where the minimum $\min_R \mathbf{T_{tot}}$ in (5) is the smallest interval according to a given ranking R in the set of intervals $\mathbb{IR}$. Constraints (6) and (7) define the tardiness of each job J_j as the interval difference between the completion time of the job and its due date. Constraints (8) and (9) establish the relationship between the starting and completion time of each task. Constraints (10) and (11) correspond to precedence relations between tasks within each job, and constraints (12) and (13) establish that the execution of two tasks requiring the same machine cannot overlap. Clearly, this problem is NP-hard, since setting all processing times and due dates to crisp numbers yields the classical JSP, which is itself NP-hard [34].

3 Robustness on Interval Schedules

At the time of scheduling, it is impossible to predict what the exact due dates and processing times will be when the project is executed. Thus, a solution to the IJSP provides an interval of all possible values for the total tardiness based on the possible values for the starting time of each task and the due date for each job. This should be understood as an a-priori or predictive solution [18]. Only when the project is actually executed, we shall know the real (deterministic) duration of each task $p_{o(j,l)} \in [\underline{p}_{o(j,l)}, \overline{p}_{o(j,l)}]$ and the final due date $d_j \in [\underline{d}_j, \overline{d}_j]$ for each job. However, a solution to the IJSP provides a task ordering π, which can be implemented in a real scenario as explained in Section 2. After execution, when processing times and due dates are exactly known, we obtain the a-posteriori solution, in which actual delays w.r.t. each job, $T_j \in [\underline{T}_j, \overline{T}_j]$ are computed.

This idea inspires the ϵ-robustness measure introduced in [32] for the job shop with fuzzy processing times and makespan minimisation, which measures the relative error of a performance metric between the a-priori and the a-posteriori solutions. In [11] we find a first proposal to adapt this measure to the interval framework and to total tardiness minimisation. It can be argued, however, that this first definition is ill-suited for the tardiness objective in the sense that it penalises the cases where the tardiness of the a-posteriori solution is lower than the predicted one. Furthermore, surpassing or staying below the predicted tardiness with the a-posteriori solution are considered equally wrong. This contradicts the idea of tardiness, a non-negative value which is worse the further it deviates from a due date and becomes null as soon a job is completed before its due date. This calls for a different robustness measure that better reflects the nature of tardiness, motivating our proposal of a new definition of robustness measure below. It relates to the proposal from [11] in the sense that they share the same inspiration based on a-priori and a-posteriori solutions, but it is crucially different in the sense that it is more in line with the idea of tardiness and it only penalises the cases where the predictive value is surpassed.

In this case, the total tardiness $\mathbf{T_{tot}}$ (an interval) is compared to the real total tardiness T_{tot}^{ex} obtained after executing a specific realization of particular task processing times and job due dates. In absence of any other information, it seems natural to estimate the total tardiness as the midpoint of $\mathbf{T_{tot}}$, $m(\mathbf{T_{tot}})$. The prediction error made by the a-priori solution can be estimated by measuring the (relative) delay of the actual executed total tardiness T_{tot}^{ex} with respect to this expected value $m(\mathbf{T_{tot}})$. We consider that given the nature of our objective function, as long as $T_{tot}^{ex} \leq m(\mathbf{T_{tot}})$ the solution can be considered robust, as the outcome is objectively better than predicted. Therefore, an a-priori solution is considered to be robust if the delay in the a-posteriori solution does not exceed the predicted one. That is, there are no "unexpected delays" in real executions. In consequence, a predictive schedule with total tardiness $\mathbf{T_{tot}}$ is *ϵ-robust* if the relative delay of the total tardiness T_{tot}^{ex} with respect to $m(\mathbf{T_{tot}})$ is less or equal than ϵ, that is:

$$R_{ex} = \frac{max(0, T_{tot}^{ex} - m(\mathbf{T_{tot}}))}{m(\mathbf{T_{tot}})} \leq \epsilon \text{ and } \epsilon \geq 0. \tag{14}$$

Clearly, the smaller the bound ϵ is, the more robust the interval schedule is.

It is quite common to use synthetic benchmark instances when real data regarding project executions are not available. In that case, K possible scenarios can be obtained using Monte-Carlo simulations [32] where deterministic values for due dates and processing times are sampled on their respective interval using uniform probability distributions. Then, the average ϵ-robustness of the predictive schedule across the K possible configurations, denoted $\bar{\epsilon}$, can be calculated as:

$$\bar{\epsilon} = \frac{1}{K}\sum_{k=1}^{K} R_k = \frac{1}{K}\sum_{k=1}^{K} \frac{max(0, T_{tot}^{k} - m(\mathbf{T_{tot}}))}{m(\mathbf{T_{tot}})}. \tag{15}$$

Algorithm 1 Schema of the Genetic Algorithm

```
Require: An IJSP instance
Ensure: A schedule
  Generate a pool $P_0$ of random solutions
  Evaluate $P_0$
  $i \leftarrow 0$
  while stopping condition is not satisfied do
    $\mathrm{Off}_i \leftarrow$ pairs of individuals selected from $P_i$
    for each pair of individuals in $\mathrm{Off}_i$ do
      Apply crossover operator with probability $p_{cross}$
      Apply mutation operator with probability $p_{mut}$
    Evaluate $\mathrm{Off}_i$
    $P_{i+1} \leftarrow$ Apply replacement operator in $(P_i, \mathrm{Off}_i)$
    $i \leftarrow i + 1$
  $Best \leftarrow$ Best solution in $P_i$ according to the interval ranking
  return $Best$
```

with T^k_{tot} denoting the deterministic total tardiness obtained after executing tasks according to the ordering π provided by the predictive schedule $\boldsymbol{s}$ on each scenario $k = 1, \ldots, K$. This value provides an estimation of how robust the schedule $\boldsymbol{s}$ is across different possible real scenarios.

4 A Genetic Algorithm for Tardiness Minimisation

Evolutionary algorithms (EAs) are a powerful tool for solving scheduling problems [19, 27]. Among them, genetic algorithms (GAs) are some of the most popular and successful, showing a superior performance either on their own or hybridised with other methods [22]. Perhaps this is the reason why, according to [9], GAs constitute the most popular approach to job shop scheduling.

In general, a GA starts by generating an initial population P_0 of individuals codifying solutions. These individuals are evaluated and assigned a fitness value each. The population is then left to evolve until a stopping criterion is met. At each iteration i, individuals from population P_i are paired for mating following a selection procedure, and recombination operators of crossover and mutation are applied to each pair with a certain probability. The new population of individuals Off_i is evaluated and a replacement operator is applied to combine P_i and Off_i into a new population P_{i+1} for the next iteration, rewarding individuals with better fitness and keeping a constant population size. Once the stopping criterion is met, the best individual from the last generation is returned. Following this template, the actual GA to be applied to solve a problem depends on the choice of solution coding schema and the genetic operators. Here we adopt the GA proposed in [11]; its pseudocode can be seen in Algorithm 1.

To encode solutions, we use classical permutations with repetition [6], where each task $o(j,l)$ is represented by its job number j. The decoding follows an insertion strategy for interval durations, which roughly consists in iterating along the chromosome and scheduling each task o(j,l) at its earliest feasible insertion position. In so doing, we always obtain a so-called active schedule in the sense that no operation can start earlier without delaying the starting time of at least another operation.

The initial population is generated at random. Different selection, crossover, mutation and replacement operators from the literature can be applied for this encoding. For selection we consider population shuffle, roulette, stochastic universal sampling and tournament 1/3 on the population. As crossover operators we take Generalised Order Crossover (GOX), Job-Order Crossover (JOX) and Precedence Preservative Crossover (PPX) and for mutation, Swap, Inversion and Insertion. Finally, for replacement we consider Generational replacement with elitism (k=1, 5%, 10%), Tournament 2/4 parents-offspring allowing repetition and Tournament 2/4 parents-offspring without repetitions. The specific choice of one operator of each kind will be decided after an empirical preliminary analysis. The stopping criterion will be having a given number of consecutive iterations without improving the best solution found so far. For more detail on the GA, we refer the interested reader to [11].

The choice of interval ranking method is also a key issue for the GA, since it affects the selection and replacement strategies, and ultimately it will decide which is the best solution to return. Therefore, in the following we shall denote by GA_{L1}, GA_{L2}, GA_{YX} and GA_{MP} the four variants of the GA obtained by considering Lex_1, Lex_2, YX and MP as corresponding ranking method.

In this setting, one may wonder if it is worth the extra computational cost of dealing with uncertainty through the optimisation process. An alternative strategy would be to transform the original problem with uncertainty into a deterministic one where task durations and due dates are taken to be the expected values of their uncertain counterparts. Therefore, we also consider another variant of the GA, GA_c, that solves this deterministic problem.

5 Experimental Results

The objective of this study is twofold: to evaluate the robustness of the schedules generated with the five variants of the proposed GA and test the sensitivity of the GA to an increasing uncertainty in data.

Tests are run on the set of instances proposed in [24], which consists in four instances (instances 1-4) with 10 jobs and 10 resources (10×10) and three instances (instances 5-7) sized 15×10. We use a PC with Intel Xeon Gold 6132 processor at 2.6 Ghz and 128 Gb RAM with Linux (CentOS v6.10) and a C++ implementation. Every variant of the algorithm is run 30 times on each instance to obtain representative data.

In [11] a parameter tuning was carried out testing several genetic operators and probabilities to find that the best setup for the GA_{MP}. This parameter tuning has been extended here to the remaining variants of the GA. The final configuration obtained for some of the parameters is common to all the variants, namely: the replacement operator, a 2/4 parents-offspring tournament without repetitions; population size equal to 250; and the stopping criterion, which consists in 25 consecutive iterations without improving the best solution found so far. As for the remaining parameters, the final configuration for each variant of the GA is shown in Table 1.

Table 1: Final parameter setup for each variant of the GA

Instance	GA_{MP}	GA_{L1}	GA_{L2}	GA_{YX}
Crossover operator	JOX	GOX	JOX	JOX
Crossover probability	1.0	0.9	1.0	1.0
Mutation operator	Insertion	Inversion	Swap	Insertion
Mutation probability	0.05	0.05	0.05	0.1
Selection operator	Shuffle	Shuffle	Shuffle	Shuffle

The GA developed in [24] (named *GA-L* hereafter) is, to the best of our knowledge, the state-of-the-art method for interval JSP with uncertain due dates. Since *GA-L* uses the $\leq_{MP}$ ranking method, the first comparison was done using GA_{MP}. Table 2 summarizes for both GA_{MP} and GA-L the total tardiness of the best-found solution together with its middle point, the average expected total tardiness across all runs, and the average CPU time in seconds In average, values GA_{MP} outperforms *GA-L* in 5 out of the 7 instances, with greater improvement on the larger instances (up to 21% better on instance 7). However, on instances 1 and 3, GA_{MP} performs worse than *GA-L*, with its best solution not even reaching the average result of *GA-L*. To better understand this surprising difference, a basic CP model of the problem was solved with the IBM CPLEX CP Optimizer solver. The solver failed to prove optimality for any instance but, noticeably, the lower bounds for the expected total tardiness obtained for the instances marked with $^{(!)}$ in Table 2 are greater than the values corresponding to the best solutions obtained by *GA-L* on those instances. This indicates that the results published for GA-L on those instances are unattainable and, hence, any comparison with them should be very cautious. In fact, the lower bound for instance 4 matches the upper bound obtained by GA_{MP}, proving the optimality of the obtained solution (see its Gantt Chart in [11]).

Once one of the variants of our *GA* has been compared against the state-of-the-art method, the five variants induced by the rankings introduced in section 2.1 are compared one to each other. A parameter tuning was carried out to find the best set up for each variant. As the objective function is an interval in all cases except GA_c, they are not directly comparable since comparisons depend on the chosen ranking. To overcome this issue, the $\bar{\epsilon}$-robustness measure is considered so the resulting solutions are compared in terms of their quality as predictive schedules. For each instance and run, each variant of

Table 2: Computational results and times of GA-L and GA_{MP}

	GA-L				GA_{MP}			
Instance	**Best**	m(**Best**)	Avg.	Time	**Best**	m(**Best**)	Avg.	Time
1	[5, 321]	163.0	**166.1**	6.5	[3, 335]	169.0	169.3	0.4
$2^{(!)}$	[0, 493]	246.5	264.0	6.6	[0, 497]	248.5	**258.6**	0.5
$3^{(!)}$	[7, 459]	233.0	**243.4**	6.4	[8, 479]	243.5	252.9	0.6
$4^{(!)}$	[4, 451]	227.5	245.2	6.3	[1, 458]	229.5	**242.3**	0.5
5	[79, 1678]	878.5	943.9	21.5	[43, 1651]	847.0	**891.9**	1.4
6	[0, 1048]	524.0	568.8	22.0	[0, 949]	474.5	**497.4**	1.4
7	[69, 1524]	796.5	999.0	21.1	[68, 1376]	722.0	**785.0**	1.2

the GA returns a predictive tardiness and a task processing order, which is then executed in $K = 1000$ deterministic realisations of the instance to calculate the $\bar{\epsilon}$ value. Table 3 shows the mean of the 30 $\bar{\epsilon}$ values multiplied by 10^3 to improve its readability. The best result in each instance is highlighted in bold. An the overwhelming difference can be appreciated between the robustness values obtained with GA_c and those obtained with any of the other four methods. Clearly, solving the associated crisp problem yields much less robust solutions than considering uncertainty during the optimisation process.

Table 3: $\bar{\epsilon}(\times 10^3)$ values for all variants of the GA

Instance	GA_c	GA_{MP}	GA_{L1}	GA_{L2}	GA_{YX}
1	389.862	65.738	64.009	**51.254**	57.876
2	338.584	3.732	3.666	**2.894**	3.626
3	368.921	3.419	**1.424**	3.534	3.666
4	352.129	5.145	6.750	**5.125**	5.277
5	169.927	44.508	72.295	**30.932**	45.868
6	815.074	0.004	**0.002**	0.014	0.010
7	168.204	**0.086**	0.384	0.811	0.296

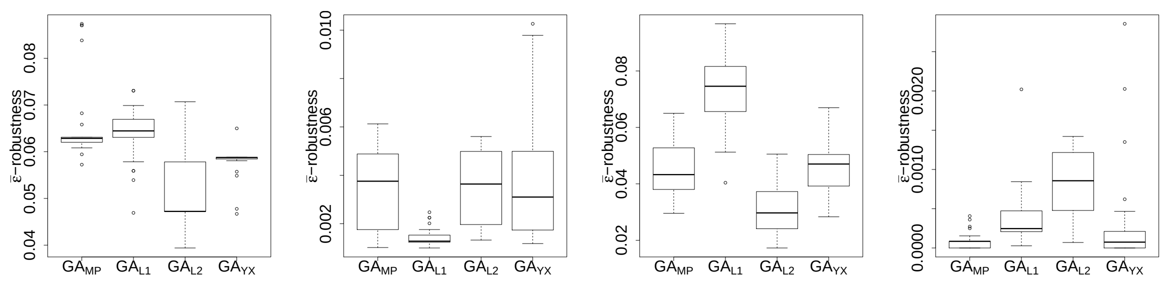
Interestingly there is not a clear winner among the interval ranking methods in terms of robustness: GA_{L2} is the best for four instances, GA_{L1} in the other two instances and GA_{MP} in the remaining one. Grey cells on Table 3 correspond to those methods for which a Kruskal-Wallis statistical test indicates that there are no significant differences w.r.t. the best solution on that instance. For a more detailed insight, Figure 1 depicts the boxplots of the 30 $\bar{\epsilon}$ values computed with each variant for those instances where the differences are clearer. These boxplots and the statistical tests confirm that there is no consensus about the most promising option in terms of $\bar{\epsilon}$-robustness and instance consistency. However, it is possible to obtain a consensus ranking by applying one of the different methods that provide a ranking of candidates based on the preferences of several voters just by considering each instance as a voter that expresses its preference over the different methods as a ranking (see Table 4) induced by the $\bar{\epsilon}$-robustness value.

Table 4: Rankings induced by the 7 instances

Instance	Ranking
1,2	$GA_{L2} \succ GA_{YX} \succ GA_{L1} \succ GA_{MP}$
3	$GA_{L1} \succ GA_{MP} \succ GA_{L2} \succ GA_{YX}$
4,5	$GA_{L2} \succ GA_{MP} \succ GA_{YX} \succ GA_{L1}$
6	$GA_{L1} \succ GA_{MP} \succ GA_{YX} \succ GA_{L2}$
7	$GA_{MP} \succ GA_{YX} \succ GA_{L1} \succ GA_{L2}$

Scoring ranking rules methods assign a score to each candidate based on its position in a ranking. A convex scoring ranking rules gives greater rewards to the candidates in better positions. The Borda count is a quite simple convex ranking rule [7] that is one of the most popular social choice functions due to its simplicity. It tries to elect broadly acceptable candidates, rather than those preferred by a majority. For this reason, it is used here to find the consensus winner GA.

For each ranking from a ranking set a candidate c_i is rewarded as many points as the number of candidates that are ranked in a position worse than c_i. The points obtained for each candidate in each ranking are then summed to obtain a final score for that candidate. The candidates are then ranked in decreasing order, so the candidate with the highest score is the winner. Applying the Borda count to the rankings in Table 4, the consensus ranking is $GA_{L2} \succ GA_{MP} \succ GA_{YX} \sim GA_{L1}$ and thus GA_{L2} is selected as the best choice.

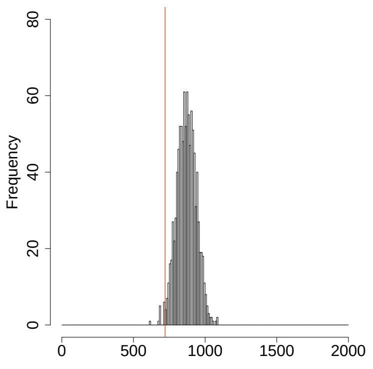


(a) Instance1 (b) Instance3 (c) Instance5 (d) Instance7

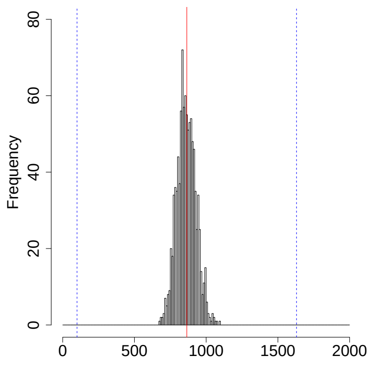
Figure 1: $\bar{\epsilon}$-robustness of schedules obtained with the different variants of GA

To illustrate the differences between GA_c and GA_{L2}, Figure 2 (subfigures (a), (b), (d) and (e)) depicts the histograms with the 1000 tardiness values obtained after executing the best schedules obtained with GA_c and GA_{L2} on the $K = 1000$ deterministic realisations of large instances 5 and 7. Instance 5 behaves similarly to the small-size instances, in the sense that the tardiness of the executed schedules does not seem to be highly affected by the choice of method. The situation is slightly different for large instances 6 and 7 where executed tardiness values seem to increase for GA_{L2}, in what appears to be the typical trade-off between robustness and objective function optimality. However, even in the

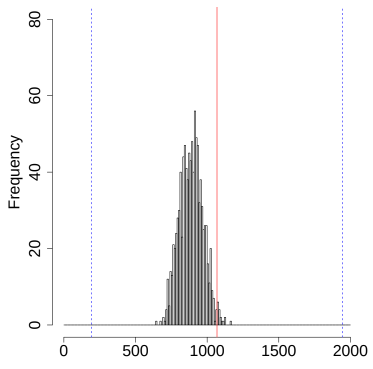
Figure 2: Histograms of T_{tot}^{ex} values of 1000 simulations for the best solutions of GA_c and GA_{L2} for instances 5 and 7, and of GA_{L2} solution for instances $5_{+40\%+20\%}$ and $7_{+40\%+20\%}$



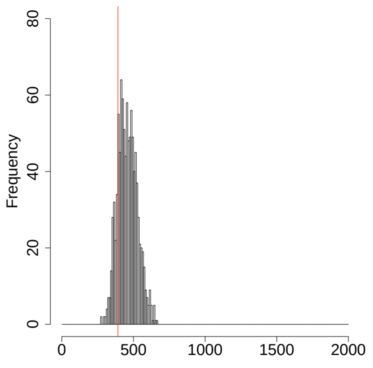
(a) Instance $5(GA_c)$



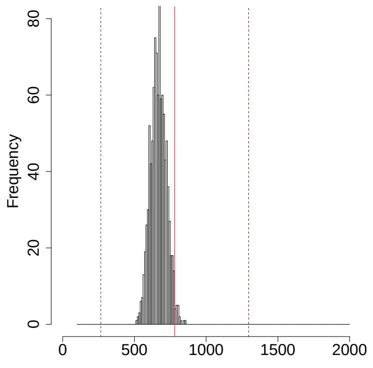
(b) Instance $5(GA_{L2})$



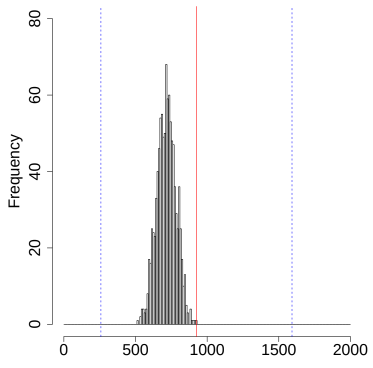
(c) Instance $5_{+40\%+20\%}(GA_{L2})$



(d) Instance $7(GA_c)$



(e) Instance $7(GA_{L2})$



(f) Instance $7_{+40\%+20\%}(GA_{L2})$

worst case, the quality loss (around 30%) is outweighed by the robustness improvement, with an increase between two and five orders of magnitude. Also, the predictive tardiness with GA_c is always lower than that with GA_{L2}. Indeed, the predictive value of GA_c tends to be on the left side of the histogram, suggesting that it is too optimistic regarding the tardiness on real executions. In other words, for the schedule obtained with GA_c there are many "unexpected" delays when it is implemented. On the other hand, the expected tardiness of the solution from GA_{L2} is more conservative and real executions are mostly below this value, thus reducing the amount of non-expected delays.

To understand better the influence of uncertainty, we perform a sensitivity analysis. Clearly, the wider an interval representing an uncertain variable, the wider the range of possible values and, hence, the greater the uncertainty. For each instance, a set of new ones with more uncertainty is generated by increasing the amplitude of the intervals representing task durations and job due dates. Taking into account the different dimension of intervals (much smaller in durations than in due dates), we have modified the range of each task duration by +20% and +40% or the amplitude of each due date by +10% and +20%. Considering the relevance of the middle point, those modifications are applied symmetrically, ensuring that there are no negative values in the lower bounds of intervals.

In total, eight variants of each original instance are created depending on whether only due dates, tasks or both are modified and on the modification rate.

Figure 2 (subfigures (c) and (f)) shows the histograms with the 1000 tardiness values obtained after executing the best schedule obtained with GA_{L2} on the variants of instances 5 and 7 that increase by 40% and 20% the amplitude of each task duration and due date respectively (denoted $5_{+40\%+20\%}$ and $7_{+40\%+20\%}$). Comparing these histograms with those in the subfigures (b) and (e), we can see that for the new instances (with greater uncertainty), the range of different executed tardiness values is wider, but the predictive tardiness is also more conservative in the sense that it is greater than the majority of the executed values. This behaviour is similar on the other instances and for every increment of uncertainty; in general, the more uncertainty is present in the instance, the more robust the obtained solutions are.

6 Conclusions

We have considered the IJSP, a version of JSP that models the uncertainty on task durations and job due dates in real-world problems using intervals. Five variants of a GA have been used to analyse the influence of the interval ranking methods on the optimisation process and also the importance of modelling the uncertainty. We have proposed a new robustness measure tailored for tardiness and we have used it to compare the different variants. Results show that when uncertainty is not modelled, the obtained solutions tend to be very optimistic, and hence much less robust. Among the different ranking methods for intervals, there seems to be no clear winner in terms of robustness, although $\leq_{MP}$ and $\leq_{YX}$ appear to be the most consistent in their results. We have also carried out a sensitivity analysis that has shown that when uncertainty in data increases, the proposed GA yields more robust solutions.

Acknowledgements

This research has been supported by the Spanish Government under research grants PID2019-106263RB-I00 and TIN2017-87600-P.

References

[1] Aissi, H., Bazgan, C., Vanderpooten, D.: Min-max and min-max regret versions of combinatorial optimization problems: A survey. European Journal of Operational Research 197, 427–438 (2009)

[2] Allahverdi, A., Aydilek, H., Aydilek, A.: Single machine scheduling problem with interval processing times to minimize mean weighted completion time. Computers & Operations Research 51, 200–207 (2014)

[3] Amjad, M.K., Butt, S.I., Kousar, R., Ahmad, R., Agha, M.H., Faping, Z., Anjum, N., Asgher, U.: Recent research trends in genetic algorithm based flexible job shop scheduling problems. Mathematical Problems in Engineering 2018, 9270802 (2018)

[4] Aytung, H., Lawley, M.A., McKay, K., Shantha, M., Uzsoy, R.: Executing production schedules in the face of uncertainties: A review and some future directions. European Journal of Operational Research 161, 86–110 (2005)

[5] Behnamian, J.: Survey on fuzzy shop scheduling. Fuzzy Optimization and Decision Making 15, 331–366 (2016)

[6] Bierwirth, C.: A generalized permutation approach to jobshop scheduling with genetic algorithms. OR Spectrum 17, 87–92 (1995)

[7] Borda, J.d.: Mémoire sur les élections au scrutin. Histoire de l'Academie Royale des Sciences (Jg. 1781) pp. 657–665 (1784)

[8] Bustince, H., Fernandez, J., Kolesárová, A., Mesiar, R.: Generation of linear orders for intervals by means of aggregation functions. Fuzzy Sets and Systems 220, 69–77 (2013)

[9] Çaliş, B., Bulkan, S.: A research survey: review of ai solution strategies of job shop scheduling problem. Journal of Intelligent Manufacturing 26(5), 961–973 (2015)

[10] Chanas, S., Kasperski, A.: On two single machine scheduling problems with fuzzy processing times and fuzzy due dates. European Journal of Operational Research 147, 281–296 (2003)

[11] Díaz, H., Palacios, J.J., Díaz, I., Vela, C.R., González-Rodríguez, I.: Tardiness minimisation for job shop scheduling with interval uncertainty. In: Hybrid Artificial Intelligent Systems. pp. 209–220. Springer (2020)

[12] Dubois, D., Prade, H., Smets, P.: Representing partial ignorance. IEEE Transactions on Systems, Man and Cybernetics, Part A 26(3), 361–377 (1996)

[13] Dubois, D., Fargier, H., Fortemps, P.: Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge. European Journal of Operational Research 147, 231–252 (2003)

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

[15] Fortin, J., Zielinski, P., Dubois, D., Fargier, H.: Criticality analysis of activity networks under interval uncertainty. Journal of Scheduling 13(6), 609–627 (2010)

[16] Garey, M., Johnson, D., Sethi, R.: The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research 1(2), 117–129 (1976)

[17] González, M., Vela, C.R., Varela, R.: A competent memetic algorithm for complex scheduling. Natural Computing 11, 151–160 (2012), 10.1007/s11047-011-9300-y

[18] González Rodríguez, I., Puente, J., Vela, C.R., Varela, R.: Semantics of schedules for the fuzzy job shop problem. IEEE Transactions on Systems, Man and Cybernetics, Part A 38(3), 655–666 (2008)

[19] Harrabi, M., Driss, O.B., Ghedira, K.: A modified biogeography-based optimization algorithm with improved mutation operator for job shop scheduling problem with time lags. Logic Journal of IGPL (2020)

[20] Kalaï, R., Lamboray, C., Vanderpooten, D.: Lexicographic α-robustness: An alternative to min-max criteria. European Journal of Operational Research 220, 722–728 (2012)

[21] Karmakar, S., Bhunia, A.K.: A comparative study of different order relations of intervals. Reliable Computing 16, 38–72 (2012)

[22] Katoch, S., Chauhan, S.S., Kumar, V.: A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications 80(5), 8091–8126 (2021)

[23] Lei, D.: Population-based neighborhood search for job shop scheduling with interval processing time. Computers & Industrial Engineering 61, 1200–1208 (2011)

[24] Lei, D.: Interval job shop scheduling problems. International Journal of Advanced Manufacturing Technology 60, 291–301 (2012)

[25] Lei, D.: Multi-objective artificial bee colony for interval job shop scheduling with flexible maintenance. International Journal of Advanced Manufacturing Technology 66, 1835–1843 (2013)

[26] Li, X., Gao, L., Wang, W., Wang, C., Wen, L.: Particle swarm optimization hybridized with genetic algorithm for uncertain integrated process planning and scheduling with interval processing time. Computers & Industrial Engineering 235, 1036–1046 (2019)

[27] Liqat, U., Vancovic, Z., Lopez-Garcia, P., Hermenegildo, H.: An evolutionary scheduling approach for trading-off accuracy vs. verifiable energy in multicore processors. Logic Journal of the IGPL 25(6), 1006–1019 (2017)

[28] Lohmer, J., Spengler, D., Lasch, R.: Multi-factory job shop scheduling with due date objective. In: Proceedings of the 2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). pp. 79–84 (2020)

[29] Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to Interval Analysis. Society for Industrial and Applied Mathematics (2009)

[30] Mou, J., Gao, L., Pan, Q., Mu, J.: Multi-objective inverse scheduling optimization of single-machine shop system with uncertain due-dates and processing times. Cluster Computing 20(1), 371–390 (2017)

[31] Palacios, J.J., González, M.A., Vela, C.R., González-Rodríguez, I., Puente, J.: Genetic tabu search for the fuzzy flexible job shop problem. Computers & Operations Research 54, 74–89 (2015)

[32] Palacios, J.J., González-Rodríguez, I., Vela, C.R., Puente, J.: Robust swarm optimisation for fuzzy open shop scheduling. Natural Computing 13(2), 145–156 (2014)

[33] Palacios, J.J., González-Rodríguez, I., Vela, C.R., Puente, J.: Coevolutionary makespan optimisation through different ranking methods for the fuzzy flexible job shop. Fuzzy Sets and Systems 278, 81–97 (2015)

[34] Pinedo, M.L.: Scheduling. Theory, Algorithms, and Systems. Springer, fifth edn. (2016)

[35] Rahmani Hosseinabadi, A.A., Vahidi, J., Saemi, B., Sangaiah, A.K., Elhoseny, M.: Extended genetic algorithm for solving open-shop scheduling problem. Soft Computing 23, 5099–5116 (2019)

[36] Roy, B.: Robustness in operational research and decision aiding: A multi-faceted issue. European Journal of Operational Research 200, 629–638 (2010)

[37] Vela, C.R., Afsar, S., Palacios, J.J., González-Rodríguez, I., Puente, J.: Evolutionary tabu search for flexible due-date satisfaction in fuzzy job shop scheduling. Computers & Operations Research 119, 104931 (2020)

[38] Xu, Z., Yager, R.R.: Some geometric aggregation operators based on intuitionistic fuzzy sets. International Journal of General Systems 35(4), 417–433 (2006)

[39] Zhang, C.Y., Li, P., Rao, Y., Guan, Z.: A very fast TS/SA algorithm for the job shop scheduling problem. Computers & Operations Research 35, 282–294 (2008)

[40] Zhang, J., Ding, G., Zou, Y., Qin, S., Fu, J.: Review of job shop scheduling research and its new perspectives under industry 4.0. Journal of Intelligent Manufacturing 30(4), 1809–1830 (2019)