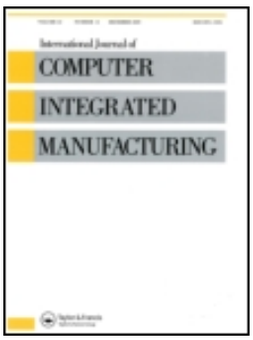
A genetic algorithm for robust hybrid flow shop scheduling

Tarek Chaari, Sondès Chaabane, Taicir Loukil, Damien Trentesaux



International Journal of Computer Integrated Manufacturing

ISSN: 0951-192X (Print) 1362-3052 (Online) Journal homepage: https://www.tandfonline.com/loi/tcim20

A genetic algorithm for robust hybrid flow shop scheduling

Tarek Chaari , Sondes Chaabane , Taicir Loukil & Damien Trentesaux

To cite this article: Tarek Chaari , Sondes Chaabane , Taicir Loukil & Damien Trentesaux (2011) A genetic algorithm for robust hybrid flow shop scheduling, International Journal of Computer Integrated Manufacturing, 24:9, 821-833, DOI: 10.1080/0951192X.2011.575181

To link to this article: https://doi.org/10.1080/0951192X.2011.575181

International Journal of Computer Integrated Manufacturing
Vol. 24, No. 9, September 2011, 821–833

Taylor & Francis
Taylor & Francis Group

A genetic algorithm for robust hybrid flow shop scheduling

Tarek Chaari[a,b,c], Sondes Chaabane[a,b]*, Taicir Loukil[d] and Damien Trentesaux[a,b]

[a]*University of Lille Nord de France, Lille, France;* [b]*UVHC, TEMPO EA 4542, F-59313 Valenciennes, France;* [c]*Faculté des Sciences Economiques et de Gestion de Sfax, Route de l'aérodrome km 4.5 B.P 1088-3018 Sfax- Tunisie;* [d]*LOGIQ, Institut Supérieur de Gestion Industrielle de Sfax, Route Mharza Km 1.5 BP 954-3018 Sfax, Tunisie*

(*Received 30 September 2010; final version received 20 March 2011*)

Most of scheduling methods consider a deterministic environment for which the data of the problem are known. Nevertheless, in reality, several kinds of uncertainties should be considered, and robust scheduling allows uncertainty to be taken into account. In this article, we consider a scheduling problem under uncertainty. Our case study is a hybrid flow shop scheduling problem, and the processing time of each job for each machine at each stage is the source of uncertainty. To solve this problem, we developed a genetic algorithm. A robust bi-objective evaluation function was defined to obtain a robust, effective solution that is only slightly sensitive to data uncertainty. This bi-objective function minimises simultaneously the makespan of the initial scenario, and the deviation between the makespan of all the disrupted scenarios and the makespan of the initial scenario. We validated our approach with a simulation in order to evaluate the quality of the robustness faced with uncertainty. The computational results show that our algorithm can generate a trade off for effectiveness and robustness for various degrees of uncertainty.

Keywords: uncertainty; robustness; effectiveness; genetic algorithm; hybrid flow shop scheduling; simulation

1. Introduction

Scheduling consists of organising, over time, the execution of tasks subjected to time and resource constraints, while satisfying one or more objectives as well as possible. Scheduling methods are abundant in the literature. The majority of these methods operate under the traditional assumption that the data in scheduling cases are perfectly known. This assumption allows problem to be treated deterministically, which considerably simplifies the solution process. However, in reality, several sources of uncertainty can have an effect on a production plan, and manufacturers are unable to provide reliable or satisfactory data for the problems that arise. The presence of uncertainty makes the question of modelling it inevitable.

This observation justified the emergence of a new theme of research, called scheduling under uncertainty. Several approaches to scheduling in uncertain environments have been proposed. These approaches include proactive approaches (also called robust approaches) (Marmier *et al*. 2009), reactive approaches (Sabuncuoglu and Kizilisik 2003) and hybrid approaches. The latter category includes two types of hybrid approaches: predictive–reactive (Yang and Geunes 2008) and proactive–reactive (Wu *et al*. 2009b).

We focus on the robust approach to scheduling defined as a proactive approach taking the uncertainty into account when designing the off-line schedule. Robust scheduling plays an essential role in various contexts, for example, in computer science, in administration, in production and also in services. Applying a meta-heuristic approach (e.g. taboo search, genetic algorithm (GA), simulated annealing) allows a good quality robust solution to be found within a reasonable time. Given the good performance of GAs in the literature for solving scheduling problems, we chose this kind of algorithm to solve a robust scheduling problem under uncertainty.

In this article, we propose a GA that is part of a robust scheduling method that takes uncertainty into account. The scenario modelling is chosen to represent the uncertainty of the problem characteristics (e.g. uncertain processing times).

Our case study is about the organisation of a hybrid flow shop. A hybrid flow shop scheduling problem (HFSSP) consists of series of production stages, each of which has several machines operating in parallel. Some stages may have multiple identical machines. All the jobs visit each stage and the order of passage through the stages is the same for all the jobs (Desprez *et al*. 2009). Each job is processed by exactly one machine at each stage, and pre-emption is not allowed. State-of-the-art survey is been proposed by Ruiz and Vázquez-Rodríguez (2010). We chose this

*Corresponding author. Email: sondes.chaabane@univ-valenciennes.fr

ISSN 0951-192X print/ISSN 1362-3052 online

DOI: 10.1080/0951192X.2011.575181
http://www.informaworld.com

case study because we are working on different applications, such as production systems and services, that could usually be considered as HFSSP (Engin *et al*. 2011). For example, we are currently working on the problem of scheduling the interventions in an operating theatre. The operating theatre scheduling problem can be modelled as an HFSSP with several stages, where the stages represent operating rooms and/or recovery rooms that are parallel and identical (Guinet and Chaabane 2003).

The rest of this article is structured as follows. In Section 2, we present a brief overview of the robust scheduling literature. In Section 3, we describe our GA for robust scheduling. In Section 4, we validate our GA for robust scheduling through simulation. Finally, in Section 5, we offer our conclusions and perspectives for future research.

2. State of the art: robust scheduling

In production scheduling, the term *robust* is often used (Shafaei and Brunn 2000, Kuroda *et al*. 2002). Jensen (2000) defines it as a term describing a solution that does not change its performance much if an uncertain parameters or unexpected events occur. Leon *et al*. (1994) define a robust schedule as one that is likely to remain valid under a wide variety of disruptions. In this article, we use the consensual definition proposed by Billaut *et al*. (2008): 'a schedule is robust if its performance is relatively insensitive to the data uncertainty'.

Various robust scheduling methods have been presented in the literature. The following are discussed below: fault-tolerance and redundancy, slack-based protection and probabilistic methods.

Fault-tolerance methods use a technique based on temporal and/or resource redundancy. With *temporal redundancy*, reserves of time (i.e. idle periods between tasks or artificially increased task durations) or protection tasks are inserted to mask machine breakdowns (Mehta and Uzsoy 1999, O'Donovan *et al*. 1999, Dolgui *et al*. 2005, Aissani *et al*. 2009).

Slack-based protection methods can be divided into two groups according to the techniques they employ: Time Window Slack (TWS) and Focused Time Window Slack (FTWS).

Instead of extending the task processing times, the TWS technique modifies the definition of the problem to insure that each task will be protected by at least a specific amount of slack time, with this slack time being included in the general time margin shared by all the tasks in the problem (Davenport *et al*. 2001).

The FTWS aims to better distribute the quantity of reserved slack time by taking the probability of breakdowns into account. If a task is sequenced late, then there is a greater probability that a disruption will occur before its execution. Therefore, this task requires more slack (Davenport *et al*. 2001).

The *probabilistic* methods allow the construction of a schedule that has the greatest probability of attaining a certain performance. Daniels and Carrillo (1997) introduced the concept of β – robustness into a single machine scheduling problem with uncertainty. The criterion they considered was the total task residence time. These authors proposed a branch-and-bound technique and heuristics that allow the construction of a schedule that maximises the probability of attaining a certain level of performance. Wu *et al*. (2009a) studied the same problem, in which the processing time of each activity was characterised by a normally distributed random variable, with the flow time as the main solution criterion. The objective is to find the β – robustness, the schedule that minimises the risk of the flow time exceeding a certain threshold. They modelled the problem as a constraint satisfaction problem.

In the area of robust scheduling, most published methods use measures to quantify schedule robustness (Sabuncuoglu and Goren 2009). Kouvelis and Yu (1997), define three robustness criteria: *absolute robustness, robust deviation* and *relative robustness* that require advance knowledge of all possible scenarios to be evaluated. Chen and Frank Chen (2008) test the robustness of the proposed flexible job shop scheduling approach under demand changes using the maximum utilisation of all machine types.

Leon *et al*. (1994) developed a robustness measure for a makespan job shop that experienced frequent disruptions in the form of breakdowns. The authors use the GA to characterise the robust solutions. The fitness function is depending on the actual makespan of the schedule during the execution and the schedule delay.

Further GA-based techniques have been described by Jensen (2001) and Sevaux and Sörensen (2004). Jensen (2001) examines the quality of schedules using a neighbourhood-based robustness measure. The initial schedule's neighbours are considered as alternative solutions if a disruption occurs.

Sevaux and Sörensen (2004) modified a GA for a single machine scheduling problem in order to find robust solutions able to deal with stochastic release dates. The authors show that the GA is able to find solutions that are both quality robust and solution robust. To allow the GA to identify robust solutions, the following robust evaluation function is used:

$$f^*(s) = \frac{1}{m}\sum_{i=1}^{m} c_i f\big(s, S_i(P)\big)$$

where f is an evaluation function for the scheduling problem, $S_i(P)$ specifies a given solution derived from

the initial one, c_i is a weight, and m is the number of possible derived solutions to evaluate.

In order to allow the simultaneous optimisation of performance and robustness, a solution robustness measure has been introduced in multi-objective evolutionary algorithms. For example, Zuo *et al*. (2009) has proposed a robust scheduling method based on a multi-objective variable neighbourhood immune algorithm in which a scheduling scheme was evaluated by the weighted sum of the average values and standard deviation of the performance criterion (makespan). Their methods focus on finding a set of Pareto robust solutions.

More recently, Goren and Sabuncuoglu (2010) proposed a proactive scheduling approach under uncertainty. They considered the single-machine scheduling problem with two sources of uncertainty: processing time variability and machine breakdowns. Two robustness measures were identified: 'expected total flow time' and 'expected total tardiness'. Three stability measures were considered: 'the sum of the squared' and 'absolute difference' of the job completion times and 'the sum of the variances' of the executed completion times. The authors proposed a beam search (BS) heuristic to generate a robust schedule. The resulting schedule was simulated 10 times, and the global evaluation function was calculated as the average of the objective values obtained. The computational results showed that the proposed BS heuristic was capable of generating stable robust schedules. Compared to a number of heuristics available in literature, their BS heuristic gave better results for all five measures of robustness and stability.

The major disadvantage of all the robustness-based techniques mentioned above is that the solution obtained is evaluated based on a set of predefined scenarios and thus does not give enough flexibility to decision makers to choose the best robust solution for their various objectives. To overcome this disadvantage, we propose a new GA for robust scheduling with a robust bi-objective evaluation function that allows an effective solution (i.e. with a minimum makespan) to be obtained that is not very sensitive to data uncertainty.

For flow shop scheduling problems, GA is one of the most frequently used techniques (Jensen 2001, Sevaux and Sörensen 2004, Desprez *et al*. 2009, Engin *et al*. 2011). The solution representation (or solution encoding), the optimisation mechanism and the evaluation function are all appropriate to solve our scheduling problem under uncertainty. Using GA to solve HFSSP allows us to maintain a set of potential solutions (i.e. populations) in each generation, which would come closer to the optimal solution generation by generation (Engin *et al*. 2011). This last property is very useful for the features of considered problems and applications of our algorithm (see Section 4.2).

3. Proposed approach

3.1. *Method/specification*

GAs are adaptive methods that can be used to solve optimisation problems (Desprez *et al*. 2009). The interested reader can consult Goldberg (1989) and Reeves (1997) for more details about the GA approach and its applications.

An appropriate representation (or encoding) is used in order to apply GA to resolve a specific problem. The solution is represented by a set of parameters known as genes joined together and referred to as chromosomes or individuals. The chromosomes define the current population. The quality of each chromosome is defined by a fitness function. At each iteration, new population is obtained from the current one using selection, crossover and mutation mechanisms. Only fittest chromosomes are selected from new and current population and used in next iteration (Honghong and Zhiming 2003, Lam *et al*. 1999).

A robust bi-objective evaluation function is presented. It allows minimising simultaneously the value of the criterion of the initial scenario and the deviation between the value of the criterion of all disrupted scenarios and the one of the initial scenario. The target of this evaluation function is to obtain an effective solution (i.e. with a minimum value of the criterion) that is not very sensitive to data uncertainty.

3.2. *GA for robust scheduling*

In this section, we present a GA for robust scheduling denoted Genetic Algorithm for Robust Scheduling (GARS). This algorithm aims to generate a robust, good-quality solution in terms of the defined criterion. To obtain a robust solution, the fitness evaluation function is replaced by a robust evaluation function. In each iteration of the GA, a set of scenarios is evaluated on all chromosomes; this set of scenarios changes from one iteration to another. In fact, this function makes it possible to increase the probability of obtaining a robust solution for a very significant number of scenarios. For example, let $Iter\_Max = 1000$ the number of total iterations of the algorithm and $N = 10$ the number of disrupted scenarios, then the robust solution is evaluated for $1000 \times 10 = 10{,}000$ cases.

3.2.1. *Robustness criterion*

The evaluation of the robust solution is based on a bi-objective robustness criterion. Let x be a solution to the problem (i.e. a permutation of the jobs). The

quality of x is computed by a robust evaluation function $f_r(x)$ (or robustness criterion). To represent the uncertainty about the characteristics of the problem, we chose a scenario modelling approach that requires constructing a set of scenarios (also called instances or data files) containing the numerical values that reflect the hypotheses about uncertainty. The uncertainty can then be modelled using discrete or continuous scenario sets, using intervals with known and certain terminals (Leung and Yue 2004).

In this article, the processing time of the jobs on each machine at each stage is considered to be uncertain. I is an initial scenario, which represents the characteristics (i.e. inputs) of the problem. The set of the scenarios, denoted by a function ξ, is represented with a sampling function that starts with the initial scenario I. Let $\xi_i(I)$ is the ith set of the sampled parameters from the initial scenario I (where $i = 1, \ldots, N$ and N represents the number of disrupted scenarios), and note that $\xi_i(I)$ are uniformly distributed between $[P_I - \alpha P_I, P_I + \alpha P_I]$, where P_I is the processing time of the initial scenario and $\alpha \in [0,1]$ is used to express the degree of uncertainty of the processing times. The uniform is a common distribution (Janak *et al.* 2007) and used to experiment our approach.

The robust evaluation function is written as follows:

$$f_r(x) = \lambda f_I(x) + (1-\lambda)\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(f_{\xi_i(I)}(x) - f_I(x)\right)^2}$$

where f_I is the value of performance criterion of initial scenario I, f_{ξ_i} is the one of disrupted scenario $\xi_i(I)$ and $\lambda \in [0,1]$ is a parameter expressing the degree of risk.

The different performance criterions (f) can be used in this function like *tardiness, makespan, total flow time*, etc.

In this article the criterion is the *makespan*. The robust evaluation function is rewritten as follows:

$$f_r(x) = \lambda C_{\max I}(x) + (1-\lambda) \times \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(C_{\max_{\xi_i(I)}}(x) - C_{\max I}(x)\right)^2}$$

This bi-objective function minimises simultaneously the makespan of the initial scenario, and the deviation between the makespan of all disrupted scenarios and the makespan of the initial scenario. The robust evaluation function is an aggregation of these two objectives. Since the two objectives do not have the same measurement scale, we must normalise them. The robust evaluation function is expressed as follows:

$$f_r(x) = \lambda \frac{C_{\max I}(x) - \mathrm{LB}}{\mathrm{LB}} + (1-\lambda) \times \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(C_{\max_{\xi_i(I)}}(x) - C_{\max I}(x)\right)^2}}{DEV\_MAX(x^*)}$$

where LB is the lower bound of initial scenario, $DEV\_MAX(x^*) = Max\{(C_{\max I}(x^*) - C_{\max I_{Min}}(x^*)), (C_{\max I_{Max}(x^*) - C_{\max I}(x^*)})\}$; (x^*) is the best solution; $C_{\max I_{Min}}(x^*)$ is the makespan of the minimal processing time of job calculated from the initial scenario. Thus, $I_{Min} = I - \alpha I$; and $C_{\max I_{Max}}(x^*)$ is the makespan of the maximal processing time of job calculated from the initial scenario ($I_{Max} = I + \alpha I$). The two values $C_{\max I_{Min}}(x^*)$ and $C_{\max I_{Max}}(x^*)$ are obtained by integrating the solution x^* obtained for the initial scenario I in the minimal and maximal scenarios.

In order to find a solution set for this problem, we varied the value of λ between 0 and 1 and applied the GARS for each value of λ.

3.2.2. *Description of our GA for robust hybrid flow shop scheduling*

We used a direct coding approach, in which a chromosome represents a schedule directly (Rajkumar and Shahabudeen 2009). We chose to maintain a simple permutation of n jobs indicating the order in which the jobs are processed in the shop at the first stage. For other stages ($k \geq 2$), the jobs are ordered according to their minimal completion time of the preceding stage (i.e. the queue of jobs in various stages ($k \geq 2$) is processed by the First In First Out (FIFO) rule). Additionally, the jobs are assigned to machines at every stage by applying the First Available Machine (FAM) assignment rule. The initial population is randomly generated. This population P_{size} is composed of a given number of chromosomes.

Selection schemes allow the algorithm to make biased decisions favouring good strings when generations change. In the literature, various selection schemes have been used (Desprez *et al.* 2009). We chose a random selection scheme in which each chromosomes has a uniform probability ($1/P_{\mathrm{size}}$) of being selected to follow the crossover process.

The crossover operator generates offspring by combining two sequences, called parents. The objective is to generate new solutions with better $C_{\max}$ values after the crossover operation. Cited by Chen *et al.* (2008): 'Murata *et al.* (1996) showed that the two-point crossover is effective for flow-shop problems'. Hence the Two-Point (TP) crossover method is used in our GA. This operator is defined as follows: two points are randomly selected for dividing one parent. The elements

outside the selected two points are always inherited from the parent to the child. The elements inside will be transferred from the second parent as they appear from left to right. Two children are produced in this way.

In this study, a shift mutation operator is used (Murata *et al*. 1996), which randomly selects a gene from the chromosome and inserts it at another random position in the chromosome.

set of disrupted scenarios according to the degree of uncertainty.

This algorithm generates a robust sequence that minimises the makespan of the initial scenario and also minimises the deviation between the makespan of all disrupted scenarios and the makespan of the initial scenario. The GARS process is illustrated as follows:

GARS Algorithm: GA for Robust Scheduling

1: **Initialise**: P_{size}, P_{cross}, P_{mut}, *Iter_Max*, $\mu = 0$, N, $\varepsilon = 0.5$, $i = 1$.
2: **Compute** *DEV_MAX*
3: **Generate** P_{size} random sequences of jobs.
4: **Repeat**
5: $\mu = 0$
6: Generate N disrupted scenarios from the initial scenario.
7: Perform the convergence test for disrupted scenarios
8: **While**
9: Randomly select two parents from the population
10: Perform a crossover operation on the two parents to obtain two children based on the probability P_{cross}.
11: Mutate the children based on the probability P_{mut}.
12: $\mu = \mu + 1$
13: **End while**
14: Evaluate all the chromosomes with a bi-objective robustness evaluation function ($2P_{size}$, parents and children).
15: Rank all chromosomes of the population $2P_{size}$ in increasing order according to the evaluation function.
16: Remove the weakest chromosomes of the population $2P_{size}$ and record the current best sequence (*Best_Pop*) and the corresponding objective function value.
17: Replace (P_{size}, *Best_Pop*)
18: $i = i + 1$
19: **Until** $i = Iter\_Max$

The fitness function plays an important role in determining the string used for the next generation. A string's fitness function is replaced by a robust evaluation function (described in Section 3.2.1).

In scheduling, *insertion* is employed to minimise the robust evaluation function. Indeed, the chromosomes which are randomly generated and the mutated children (equal to $2 \times P_{\text{size}}$) are sorted in an increasing order according to their fitness function. Only the higher half of the population, corresponding to the best chromosomes, is selected for the next iteration. Thus, the size of the population remains constant (equal to P_{size}) from generation to generation. The stopping criterion is a fixed number of generations (or iterations) equal to *Iter _ Max*.

The GARS considers the initial scenario. Using this initial scenario, for each iteration of the GA, it builds a

3.2.3. Convergence test

The convergence test allows the number of disrupted scenarios to be determined. There are several techniques to test the convergence and thus determine the number of scenarios (e.g. average, standard deviation). For this article, the average of all the disrupted scenarios is used.

In our GARS, the number of disrupted scenarios is $N \in [10, 20, \ldots, 100]$ and $\varepsilon = 0.5$. If the average $C_{\max}$ of N disrupted scenarios minus the average $C_{\max}$ of $N + 10$ disrupted scenarios is lower than ε, the convergence test is satisfied:

$$\frac{1}{N}\sum_{i=1}^{N} C_{\max\xi_i(I)}(x) - \frac{1}{N+10}\sum_{i=1}^{N+10} C_{\max\xi_i(I)}(x) \leq \varepsilon\,,$$

in this case the number of disrupted scenarios is fixed at N.

If the convergence test is not satisfied then this test is stopped and the number of disrupted scenarios is set at $N = 100$.

4. Validation of GARS using simulation

We validated the GARS using simulation to highlight and measure its effectiveness at generating robust solutions.

Our GARS is implemented in C++ on a Pentium IV with 1.7GHz and 512 Mo of RAM. We conducted several experimental tests to choose the best parameters values for our GAs. We retained the following values:

- Population size $P_{\text{size}} = 200$;
- Crossover probability $P_{\text{cross}} = 0.9$;
- Mutation probability $P_{\text{mut}} = 0.2$;
- Stopping criterion $Iter\_Max = 1000$ generations.

4.1. Description of the experimental approach

We used the ARENA 12.0 simulation software to create a simulation model, which allowed us to test and validate the robustness obtained by GARS. The (Figure 1) presents our experimental approach.

The simulation model corresponds to a possible organisation of a hybrid flow shop with five production stages and is briefly described in below. The GARS allows a *robust sequence* to be calculated, which minimises the robust evaluation function ($f_r(x)$). As input, the simulation receives the sequence of the jobs obtained by GARS. We introduced stochastic data into every simulated scenario in the model. These data were randomly generated according to the uniform distribution between $[P_I - \alpha P_I,\ P_I + \alpha P_I]$, such as defined in Section 3.2.1. (i.e. P_I the processing time of the initial scenario and $\alpha \in [0,1]$ is used to express the degree of uncertainty of the processing times).

N_R, ($N_R = 100$) is a number of replications done to simulate each obtained robust sequence.

The assignment rules for the jobs in the various stages are the same as the assignment rules used in the GARS (i.e. the simulator uses the robust scheduling for the arrival order of jobs only in the first stage, for the other stages the FIFO rule is applied). The assignment of jobs to machines at every stage applies the FAM assignment rule.

4.2. Experimental results and discussions

In this experiment, we used the benchmark problems of Carlier and Neron (2000) for each configuration type, and we randomly generated other tests in which the size varies between 20 jobs and 100 jobs in two stages. Table 1 presents the configurations of these benchmark problems. The column 'Benchmark' gives the name of the instance. The stage number and job number are given respectively in second and third columns. The column 'Machine configuration in each stage' presents the number of machines in each stage. For example, '2 2 1 2 2' is two machines in stages 1, 2, 4 and 5 and only one machine in stage 3. The last column is the processing time intervals [*Min, Max*].

Several values of α were tested for each problem and were varied between 10% and 50%. For each degree of uncertainty α and for each degree of risk λ, the GARS was run to obtain a sequence. Once this sequence was obtained, the simulator received this sequence of the jobs as input. We evaluated 100 replications of the problem instance using this sequence and based on stochastic data as defined in Section 4.1.

Tables 2, 3, and 4 give the detailed results for the GARS for the set of instances with a degree of uncertainty equal 10%, 25% and 50%, respectively. In these tables, the column 'Instance' gives the number of the problem instance. The column 'LB' gives the lower bound of the initial scenario I (note that we use the lower bound developed by Santos *et al.* (1995) for the

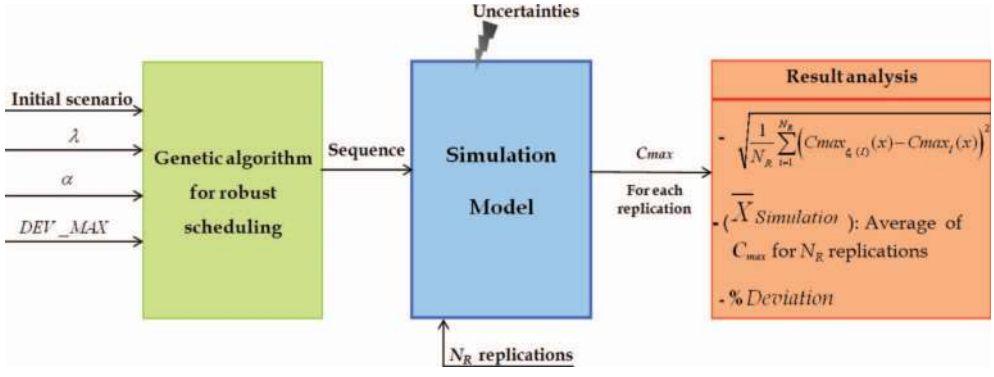


Figure 1. Experimental approach.

Table 1. The benchmark problems used in our experiments.

Benchmark	Stage number	Job number	Machine configuration in each stage	Processing time between
j10c5a2	5	10	2 2 1 2 2	[3, 20]
j10c5b1	5	10	1 2 2 2 2	[3, 20]
j10c5c1	5	10	3 3 2 3 3	[3, 20]
j10c5d1	5	10	3 3 3 3 3	[3, 20]
j15c5a1	5	15	2 2 1 2 2	[3, 20]
j15c5b1	5	15	1 2 2 2 2	[3, 20]
j15c5c1	5	15	3 3 2 3 3	[3, 20]
j15c5d1	5	15	3 3 2 3 1	[3, 20]
2center20job	2	20	2 2	[5, 25]
2center50job	2	50	3 3	[5, 25]
2center100job	2	100	4 4	[5, 25]

Table 2. Experimental results for the degree of uncertainty $\alpha = 10\%$.

Instance	LB	λ	$C_{\max I}$	N	CPU(s)	$\sqrt{\frac{1}{N_R}\sum_{i=1}^{N_R}\left(C_{\max\xi_i(I)}(x) - C_{\max I}(x)\right)^2}$	$\overline{X}_{\text{simulation}}$ (Average $C_{\max}$ for 100 replications)	% Deviation of $\overline{X}_{\text{simulation}}$ from $C_{\max I}$ in (%)
J10c5a2	88	$\lambda = 1$	88	–	0.02	8.83	96.04	9.13
		$\lambda = 0.5$	93	10	295.10	4.24	96.29	3.53
		$\lambda = 0$	124	20	22.81	3.73	124.83	0.66
J10c5b1	130	$\lambda = 1$	130	–	0.01	7.06	136.66	5.12
		$\lambda = 0.5$	134	10	374.45	4.34	137.42	2.55
		$\lambda = 0$	146	20	189.71	4.01	148.1	1.43
J10c5c1	68	$\lambda = 1$	68	–	0.05	7.80	75.37	10.83
		$\lambda = 0.5$	76	10	210.04	3.60	78.64	3.47
		$\lambda = 0$	89	10	52.78	2.80	89.39	0.43
J10c5d1	66	$\lambda = 1$	66	–	0.03	7.92	73.53	11.40
		$\lambda = 0.5$	72	10	15.67	2.70	73.01	1.40
		$\lambda = 0$	79	20	148.36	1.91	79.74	0.93
J15c5a1	178	$\lambda = 1$	178	–	0.02	9.30	186.48	4.76
		$\lambda = 0.5$	185	10	26.93	5.13	188.6	1.94
		$\lambda = 0$	205	20	319.04	5.07	205.2	0.09
J15c5b1	170	$\lambda = 1$	170	–	0.02	9.86	179.31	5.47
		$\lambda = 0.5$	175	10	580.31	6.82	181.05	3.45
		$\lambda = 0$	188	30	466.17	4.79	190.47	1.31
J15c5c1	85	$\lambda = 1$	85	–	0.08	10.82	95.47	12.31
		$\lambda = 0.5$	94	10	417	3.85	95.92	2.04
		$\lambda = 0$	114	10	94.15	2.83	113.61	−0.34
J15c5d1	167	$\lambda = 1$	167	–	0.03	8.63	175.08	4.83
		$\lambda = 0.5$	174	10	156.18	6.04	179.1	2.93
		$\lambda = 0$	183	20	493.82	4.71	184.22	0.66
2centre20job	161	$\lambda = 1$	161	–	0.02	8.97	169.23	5.11
		$\lambda = 0.5$	165	10	313.18	4.04	168.3	2.00
		$\lambda = 0$	177	30	158.54	3.93	175.23	−1.00
2centre50job	256	$\lambda = 1$	256	–	0.2	10.98	266.75	4.19
		$\lambda = 0.5$	269	10	211.53	4.66	272.97	1.47
		$\lambda = 0$	298	10	156.32	4.27	298.29	0.09
2centre100job	402	$\lambda = 1$	402	–	2.2	15.15	416.92	3.71
		$\lambda = 0.5$	417	10	970.07	5.86	421.82	1.15
		$\lambda = 0$	433	10	1107.31	4.96	435.74	0.63

test problems randomly generated). The column 'λ' gives the degree of risk (or relative importance) given to the two objectives. The column '$C_{\max I}$' gives the makespan value of the initial scenario obtained by GARS. The column 'N 'gives the number of disrupted scenarios in each iteration of GARS. The column 'CPU(s)' gives the CPU time of the GARS in seconds. The deviation between the makespan of all disrupted scenarios and the makespan of the initial scenario is reported in the column '$\sqrt{\frac{1}{N_R}\sum_{i=1}^{N_R}\left(C_{\max\xi_i(I)} - C_{\max I}\right)^2}$,'

Table 3. Experimental results for the degree of uncertainty $\alpha = 25\%$.

Instance	LB	λ	$C_{\max I}$	N	CPU(s)	$\sqrt{\frac{1}{N_R}\sum_{i=1}^{N_R}\left(C_{\max\xi_i(I)}(x) - C_{\max I(x)}\right)^2}$	$\overline{X}_{\text{simulation}}$ (Average $C_{\max}$ for 100 replications)	% Deviation of $\overline{X}_{\text{simulation}}$ from $C_{\max I}$ in (%)
J10c5a2	88	$\lambda = 1$	88	–	0.02	9.63	96.19	9.30
		$\lambda = 0.5$	93	30	61.81	5.36	96.07	3.30
		$\lambda = 0$	109	20	10.04	4.18	107.85	−1.05
J10c5b1	130	$\lambda = 1$	130	–	0.01	7.10	135.4	4.15
		$\lambda = 0.5$	134	10	211.12	5.51	136.21	1.64
		$\lambda = 0$	139	40	323.18	5.43	140.89	1.35
J10c5c1	68	$\lambda = 1$	68	–	0.05	9.88	77.14	13.44
		$\lambda = 0.5$	75	60	274.15	4.25	77.56	3.41
		$\lambda = 0$	84	30	229.82	3.87	83.6	−0.47
J10c5d1	66	$\lambda = 1$	66	–	0.03	8.07	73.39	11.19
		$\lambda = 0.5$	72	10	80.57	3.42	72.21	0.29
		$\lambda = 0$	78	30	125.26	2.83	76.81	−1.52
J15c5a1	178	$\lambda = 1$	178	–	0.02	12.78	188.32	5.79
		$\lambda = 0.5$	183	20	241.79	9.28	188.88	3.21
		$\lambda = 0$	208	40	143.71	8.42	209.74	0.8
J15c5b1	170	$\lambda = 1$	170	–	0.02	9.36	177.41	4.35
		$\lambda = 0.5$	171	50	247.01	7.68	176.05	2.95
		$\lambda = 0$	182	40	7.54	6.96	185.06	1.68
J15c5c1	85	$\lambda = 1$	85	–	0.08	11.95	96.33	13.32
		$\lambda = 0.5$	93	10	24.10	3.68	95.62	2.81
		$\lambda = 0$	102	40	80.84	3.48	102.26	0.25
J15c5d1	167	$\lambda = 1$	167	–	0.03	9.31	174.25	4.34
		$\lambda = 0.5$	172	20	681.28	6.57	175.07	1.78
		$\lambda = 0$	180	30	8.79	5.98	180.58	0.32
2centre20job	161	$\lambda = 1$	161	–	0.02	11.00	170.02	5.60
		$\lambda = 0.5$	164	30	299.96	6.27	167.72	2.26
		$\lambda = 0$	178	40	268.06	6.18	178.76	0.42
2centre50job	256	$\lambda = 1$	256	–	0.18	13.50	268.6	4.92
		$\lambda = 0.5$	265	40	252.46	6.89	269.14	1.56
		$\lambda = 0$	284	30	964.51	5.90	286.04	0.71
2centre100job	402	$\lambda = 1$	402	–	2.2	16.50	417.64	3.89
		$\lambda = 0.5$	415	30	887.29	8.64	421.4	1.54
		$\lambda = 0$	441	20	57.06	7.68	442.02	0.23

The column '$\overline{X}_{\text{simulation}}$' presents the average of $C_{\max}$ value for 100 replications. The column '% Deviation' (where $\%\,\text{Deviation} = \frac{\bar{X}_{\text{simulation}} - C_{\max I}}{C_{\max I}}$) gives the deviation or the $C_{\max}$ increase (in %) of the 100 disrupted scenarios compared to the initial scenario's makespan. For example, for the instance, j10c5a2, if only the makespan of the initial scenario is minimised, modifying the processing time with a degree of uncertainty equal to 10% could lead to an average increase of 9.13% in the makespan. But if only the deviation is minimised, modifying the processing time with the same degree of uncertainty could lead to an average increase of 0.66% in the makespan.

For $\lambda = 1$, we recommend minimising the makespan of the initial scenario (note that the disrupted scenarios do not appear in algorithm).

For $\lambda = 0$, we recommend minimising the deviation between the makespan of all disrupted scenarios and the makespan of the initial scenario $\left(\sqrt{\frac{1}{N_R}\sum_{i=1}^{N_R}\left(C_{\max\xi_i(I)} - C_{\max I}\right)^2}\right)$. According to Tables 2, 3, and 4, the more λ decreases, the more the makespan of the initial scenario increases and the more the deviation between the makespan of all disrupted scenarios and the makespan of the initial scenario decreases.

Let us examine in detail the instance j10c5a2. The makespan of the initial scenario obtained using $\lambda = 1$ is 88, which is close (or equal) to the LB. If we maintain this sequence, knowing that processing time can be disrupted with degree of uncertainty equal to 10%, we could expect an average makespan of 96.04 after the simulation, which is a cost increase of 9.13%. Therefore, this solution is sensitive to uncertainty but gives good results for $C_{\max}$.

For the same instance, if $\lambda = 0$, the GARS generates a sequence with a solution of 124, which is far from LB. If this sequence is used in the disrupted scenarios, the expected makespan value of the initial scenario will be an average of 124.83, which is an

Table 4. Experimental results for the degree of uncertainty $\alpha = 50\%$.

Instance	LB	λ	$C_{\max I}$	N	CPU(s)	$\sqrt{\frac{1}{N_R}\sum_{i=1}^{N_R}(C_{\max\xi_i(I)}(x) - C_{\max I}(x))^2}$	$\overline{X}_{\text{simulation}}$ (Average $C_{\max}$ for 100 replications)	%Deviation of $\overline{X}_{\text{simulation}}$ from $C_{\max I}$ in (%)
J10c5a2	88	$\lambda = 1$	88	–	0.02	13.12	98.9	12.38
		$\lambda = 0.5$	93	60	95.95	8.34	97.52	4.86
		$\lambda = 0$	111	40	26	8.15	109.95	−0.94
J10c5b1	130	$\lambda = 1$	130	–	0.01	11.50	136.12	4.70
		$\lambda = 0.5$	130	30	38.64	10.03	135.17	3.97
		$\lambda = 0$	137	40	134.82	9.85	138.81	1.32
J10c5c1	68	$\lambda = 1$	68	–	0.05	13.12	79.76	17.29
		$\lambda = 0.5$	73	40	405.14	7.28	77.61	6.31
		$\lambda = 0$	79	10	264.32	6.14	80.66	2.10
J10c5d1	66	$\lambda = 1$	66	–	0.03	10.06	74.37	12.68
		$\lambda = 0.5$	72	30	330.18	5.28	73.84	2.55
		$\lambda = 0$	79	20	137.21	5.02	78.31	−0.87
J15c5a1	178	$\lambda = 1$	178	–	0.02	16.52	186.86	4.97
		$\lambda = 0.5$	178	30	558.06	14.76	184.8	3.82
		$\lambda = 0$	186	20	467.11	13.46	187.97	1.05
J15c5b1	170	$\lambda = 1$	170	–	0.02	13.18	177.51	4.41
		$\lambda = 0.5$	170	40	195.85	12.24	175.9	3.47
		$\lambda = 0$	182	30	512.17	11.71	183.71	0.93
J15c5c1	85	$\lambda = 1$	85	–	0.08	13.78	97.68	14.91
		$\lambda = 0.5$	91	40	64.37	7.29	96.35	5.87
		$\lambda = 0$	100	40	52.59	6.76	99.93	−0.07
J15c5d1	167	$\lambda = 1$	167	–	0.03	13.36	172.15	3.08
		$\lambda = 0.5$	167	10	507.43	11.81	169.83	1.69
		$\lambda = 0$	173	20	130.62	11.43	174.12	0.64
2centre20job	161	$\lambda = 1$	161	–	0.02	12.59	168.98	4.95
		$\lambda = 0.5$	162	20	289.15	10.51	167.61	3.46
		$\lambda = 0$	169	20	322.03	9.37	169.52	0.30
2centre50job	256	$\lambda = 1$	256	–	0.2	17.44	270.99	5.85
		$\lambda = 0.5$	264	20	250.68	10.47	270.21	2.35
		$\lambda = 0$	273	60	362.54	10.07	277.49	1.64
2centre100job	402	$\lambda = 1$	402	–	2.14	19.92	418.71	4.15
		$\lambda = 0.5$	405	50	867.64	15.98	414.83	2.42
		$\lambda = 0$	432	30	946.86	11.98	434.99	0.69

increase of only 0.66%. The solution given by the GARS is then more robust in terms of managing uncertainty but not effective in terms of makespan.

If we look for a trade off between effectiveness and scheduling robustness, the value of λ will depend on the viewpoint of the decision maker. In our case of $\lambda = 0.5$, the GARS generates a sequence with makespan of the initial scenario equal to 93. After the simulation, we expect an average makespan of 96.29, which is a cost increase of 3.53%. This solution is more robust than the solution found for $\lambda = 1$ and more effective than one for $\lambda = 0$.

According to the tables for the different degrees of uncertainty, the '% Deviation' decreases when λ decreases. This decrease means that more confidence should be granted to the robust solution even if the makespan of the initial scenario is higher. In other words, our initial assumption is confirmed: non-robust schedules will rapidly become ineffective with uncertainty, while robust scheduling will not deteriorate as fast as non-robust schedules do when uncertainty is introduced.

When the degree of uncertainty α increases from 10% to 25% then $\lambda = 0.5$ will provide in most cases similar solution to $\lambda = 1$, however, in some cases the solution is better. This allows us to conclude that our approach is more advantageous with higher degrees of uncertainty. Figures 2, 3, and 4 confirm our conclusion. They present the results of 100 simulated replications for the instance j10c5a2 with degrees of uncertainty equal to 10%, 25% and 50%.

Table 5 provides the summarised results for the whole set of instances for the different degrees of uncertainty. As shown in Table 5, for the % Deviation, the gap between $\lambda = 1$ and $\lambda = 0.5$ is more important than the gap between $\lambda = 0.5$ and $\lambda = 0$.

Nevertheless, the CPU times of GARS (266.18 s in case $\lambda = 0$ and 316.2 s in case $\lambda = 0.5$) are much higher than the CPU times in case $\lambda = 1$ because the calculations of the bi-objective robust evaluation

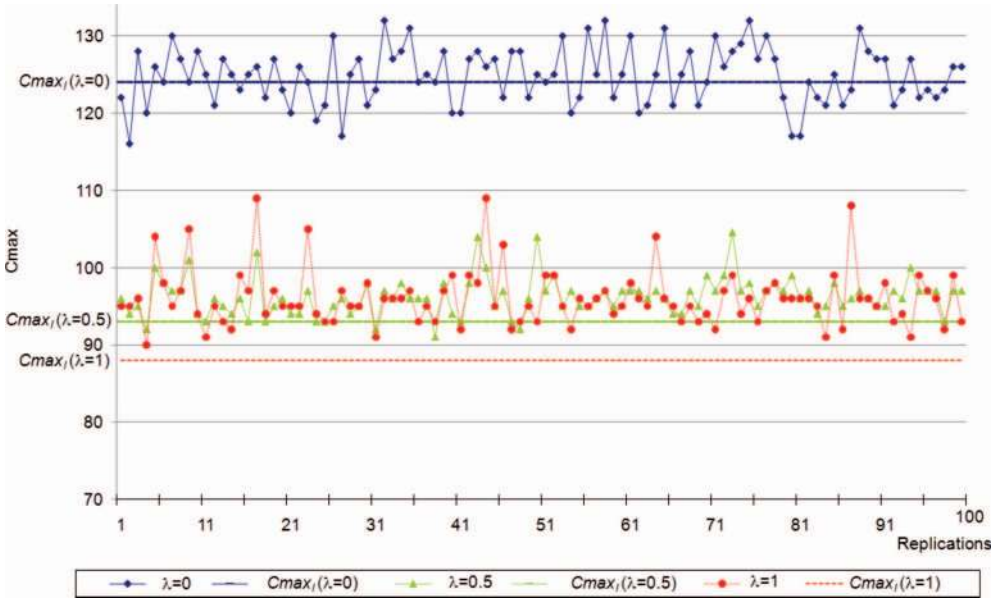


Figure 2. Example of 100 replications for the instance j10c5a2 with a degree of uncertainty $\alpha = 10\%$.

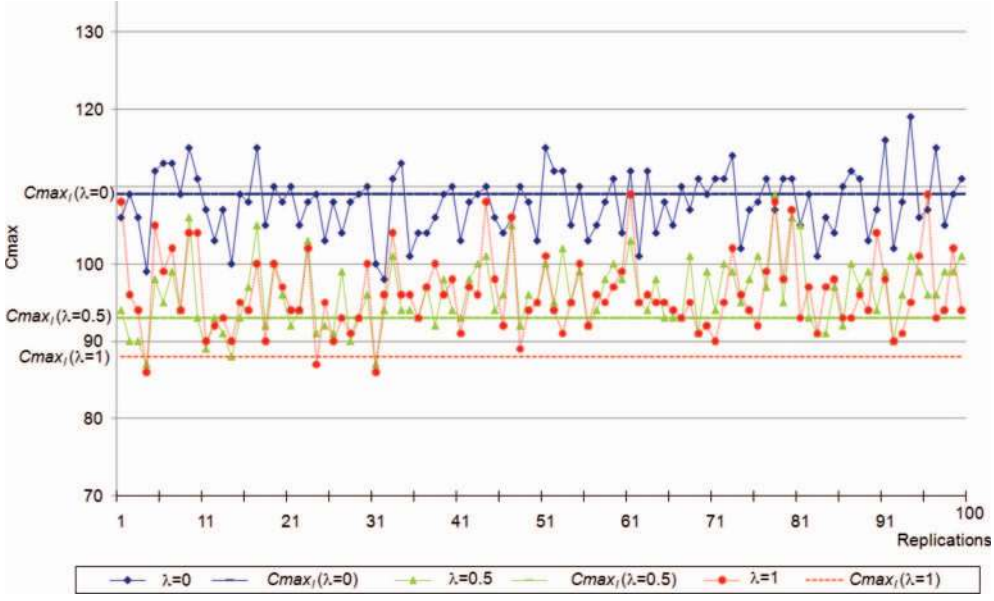


Figure 3. Example of 100 replications for the instance j10c5a2 with a degree of uncertainty $\alpha = 25\%$.

function requires a high number of scenarios in each iteration.

As mentioned in the introduction of this article, one of the applications of our GARS concerns the operating theatre scheduling problem. This problem can be modelled as a HFSSP with several stages, where the stages represent operating rooms and/or recovery rooms that are parallel and identical (Guinet and Chaabane 2003). The managers must take into account the variation of the surgery durations to prevent future disruptions when generating the off-line schedule. Their goal is thus to produce a *robust* schedule as insensitive as possible to the variation of the surgery duration. This variation can be modelled probabilistically using distribution laws. Our GA is used as a first step towards a decision-support system since it already helps operating theatre managers by providing an off-line schedule that respects the overall traditional operational constraints (e.g. capacity, opening hours). In a second step, a user-friendly interface is

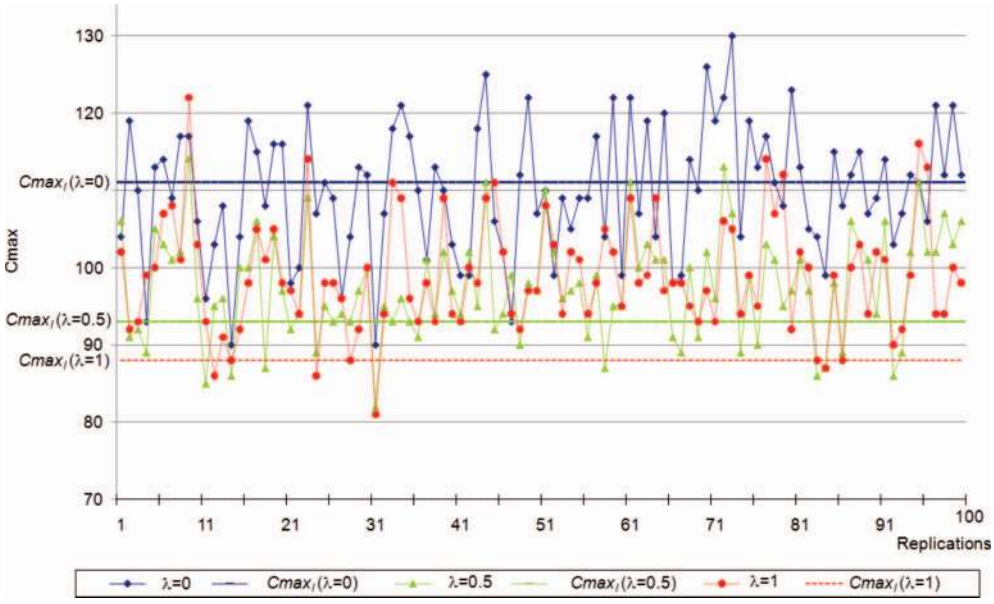


Figure 4. Example of 100 replications for the instance j10c5a2 with a degree of uncertainty $\alpha = 50\%$.

Table 5. The average % deviation for all problems.

	$\alpha = 10\%$	$\alpha = 25\%$	$\alpha = 50\%$
$\lambda = 1$	7.31	7.64	8.52
$\lambda = 0.5$	2.47	2.32	3.78
$\lambda = 0$	0.42	0.24	0.61

being developed to allow the managers to adapt the schedule obtained by integrating unmodelled constraints, such as surgeon preferences and surgical team affinity.

A second application of our GARS was also used in a flexible manufacturing system (Sallez *et al*. 2010). The problem was identified as a flexible job-shop scheduling problem with transportation times and a limited number of conveyor shuttles constraints. This problem can be considered as an extension of HFSSP (Guinet 2000). The idea is to embed a real-time version of our GARS into products to calculate an effective, robust real-time distributed schedule according to the real state of the production system. The GA is always able to propose solutions, whatever time constraints, and this ability is very interesting in this context (Honghong and Zhiming 2003). (Of course, the more time a product has to decide, the better the solution proposed by the GA should be.)

5. Conclusion

In this article, we proposed a GARS in HFSSP with uncertain processing times. In this algorithm, we integrated a new mechanism for finding robust and effective schedules, which makes it possible to increase the probability of obtaining a robust solution on a very significant number of scenarios. We also added a new robust evaluation function, which is bi-objective. It minimises simultaneously the makespan of the initial scenario to obtain an effective solution, and the deviation between the makespan of all disrupted scenarios and the makespan of the initial scenario in order to obtain a robust solution.

Our GARS was validated through simulation. The simulation results show that the proposed algorithm can generate a trade off for effectiveness and robustness for a very high degree of uncertainty. Our algorithm is not really relevant without any degree of uncertainty, but it becomes increasingly relevant and reliable with growing degrees of uncertainty. In fact, it helps to measure quantitatively the impact of uncertainty and thus to manage the risk induced by such uncertainty.

Several future avenues of research are planned. The first is to implement a change aggregation function using a multi-criterion analysis mechanism to consider the bi-objective or multi-objective problems. The second is to implement a new mechanism to evaluate a set of scenarios for each evaluation of a single chromosome. The third is to incorporate our GARS in methodological approach based on simulation-optimisation techniques for sizing manufacturing systems. We are currently working in this direction, trying to apply this method to the designing and sizing of operating theatres.

Acknowledgements

This research was partially financed by the French Ministry of Higher Education and Scientific Research and the Ministry of Foreign Affairs (project PHC/EGIDE/UTIQUE/09G1402). We are grateful for the support of these institutions.

References

Aissani, N., Beldjilali, B., and Trentesaux, D., 2009. Dynamic scheduling of maintenance tasks in the petroleum industry: a reinforcement approach. *Engineering Applications of Artificial Intelligence*, 22 (7), 1089–1103.

Billaut, J.-C., Moukrim, A., and Sanlaville, E., 2008. *Flexibility and robustness in scheduling*. London: John Wiley & Sons.

Carlier, J. and Neron, E., 2000. An exact method for solving the multiprocessor flowshop. *RAIRO Operations Research,* 34*, l–25*.

Chen, J. and Frank Chen, F., 2008. Adaptive scheduling and tool flow control in flexible job shops. *International Journal of Production Research*, 46 (15), 4035–4059.

Chen, J.-S., Chao-Hsien Pan, J., and Lin, C.-M., 2008. A hybrid genetic algorithm for the re-entrant flow-shop scheduling problem. *Expert Systems with Applications*, 34 (1), 570–577.

Daniels, R.L. and Carrillo, J.E., 1997. β-Robust scheduling for single machine systems with uncertain processing times. *IIE Transactions*, 29, 977–985.

Davenport, A., Gefflot, C., and Beck, J., 2001. Slack-based techniques for robust schedules. *In: Sixth European Conference on Planning (ECP–2001)*. Toledo, Spain. Berlin: Springer Verlag.

Desprez, C.C., Chu, F.F., and Chu, C.C., 2009. Minimising the weighted number of tardy jobs in a hybrid flow shop with genetic algorithm. *International Journal of Computer Integrated Manufacturing*, 22 (8), 745–757.

Dolgui, A.A., Levin, G.G., and Louly, M.A., 2005. Decomposition approach for a problem of lot-sizing and sequencing under uncertainties. *International Journal of Computer Integrated Manufacturing*, 18 (5), 376–385.

Engin, E., Ceran, G., and Yilmaz, M.K., 2011. An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems. *Applied Soft Computing*, 11 (3), 3056–3065.

Goldberg, D.E., 1989. *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Goren, S. and Sabuncuoglu, I., 2010. Optimization of schedule robustness or stability under random machine breakdowns and processing time variability. *IIE Transactions*, 42 (3), 203–220.

Guinet, A., 2000. Efficiency of reductions of job-shop to flow-shop problems. *European Journal of Operational Research*, 125 (3), 469–485.

Guinet, A. and Chaabane, S., 2003. Operating theatre planning. *International Journal of Production Economics*, 85 (1), 69–81.

Honghong, Y. and Zhiming, W., 2003. The application of Adaptive Genetic Algorithms in FMS dynamic rescheduling. *International Journal of Computer Integrated Manufacturing*, 16 (6), 382.

Janak, S.L., Lin, X., and Floudas, C.A., 2007. A new robust optimization approach for scheduling under uncertainty: II. Uncertainty with known probability distribution. *Computers & Chemical Engineering*, 31 (3), 171–195.

Jensen, M., 2000. Neighbourhood based robustness applied to tardiness and total flowtime jobshops. *In*: *Proceeding of Sixth Parallel Problem Solving from Nature,* 1917. Paris, France. Berlin: Springer, 283–292.

Jensen, M.T., 2001. Improving robustness and flexibility of tardiness and total flow-time job shops using robustness measures. *Applied Soft Computing*, 1 (1), 35–52.

Kouvelis, P. and Yu, G., 1997. *Robust discrete optimisation and its applications*. London: Kluwer Academic Publisher.

Kuroda, M., Shin, H., and Zinnohara, A., 2002. Robust scheduling in an advanced planning and scheduling environment. *International Journal of Production Research*, 40 (15), 3655–3668.

Lam, F.S.C., *et al*., 1999. Scheduling to minimize product design time using a genetic algorithm. *International Journal of Production Research*, 37 (6), 1369–1386.

Leon, V., Wu, S., and Storer, R., 1994. Robustness measures and robust scheduling for job shops. *IIE Transactions*, 26 (5), 32–43.

Leung, S. and Yue, W., 2004. A robust optimization model for stochastic aggregate production planning. *Production Planning & Control*, 15 (5), 502–514.

Marmier, F., Varnier, C., and Zerhouni, N., 2009. Proactive, dynamic and multi-criteria scheduling of maintenance activities. *International Journal of Production Research*, 47 (8), 2185–2201.

Mehta, S.V. and Uzsoy, R.H., 1999. Predictive scheduling of a single machine subject to breakdowns. *International Journal of Computer Integrated Manufacturing*, 12 (1), 15–38.

Murata, T., Ishibuchi, H., and Tanaka, H., 1996. Genetic algorithms for flows shop scheduling problems. *Computers & Industrial Engineering*, 30 (4), 1061–1071.

O'Donovan, R., Uzsoy, R., and McKay, K.N., 1999. Predictable scheduling of a single machine with breakdowns and sensitive jobs. *International Journal of Production Research*, 37 (18), 4217–4233.

Rajkumar, R.R. and Shahabudeen, P.P., 2009. Bi-criteria improved genetic algorithm for scheduling in flowshops to minimise makespan and total flowtime of jobs. *International Journal of Computer Integrated Manufacturing*, 22 (10), 987–998.

Reeves, C.R., 1997. Genetic algorithms for the operations researcher. *INFORMS Journal on Computing*, 9, 231–250.

Ruiz, R. and Vázquez-Rodríguez, J.A., 2010. The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 205, 1–18.

Sabuncuoglu, I. and Goren, S., 2009. Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research. *International Journal of Computer Integrated Manufacturing*, 22 (2), 138–157.

Sabuncuoglu, I. and Kizilisik, O.B., 2003. Reactive scheduling in a dynamic and stochastic FMS environment. *International Journal of Production Research*, 41 (17), 4211–4231.

Sallez, Y., *et al*., 2010. Semi-heterarchical control of FMS: from theory to application. *Engineering Applications of Artificial Intelligence*, 23 (8), 1314–1326.

Santos, D.L., Hunsucker, J.L., and Deal, D.E., 1995. Global lower bounds for flow shops with multiple processors. *European Journal of Operational Research*, 80, 112–120.

Sevaux, M. and Sörensen, K., 2004. A genetic algorithm for robust schedules in a just-in-time environment with ready times and due dates. *4OR - Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 2 (2), 129–147.

Shafaei, R. and Brunn, P., 2000. Workshop scheduling using practical (inaccurate) data Part 3: A framework to integrate job releasing, routing and scheduling functions to create a robust predictive schedule. *International Journal of Production Research*, 38 (1), 85–99.

Wu, C.W., Brown, K.N., and Beck, J.C., 2009a. Scheduling with uncertain durations: modeling beta-robust scheduling with constraints. *Computers and Operations Research*, 36, 2348–2356.

Wu, L.H., *et al*., 2009b. The research on proactive-reactive scheduling framework based on real-time manufacturing information. *Materials Science Forum*, 626–627, 789–794.

Yang, B. and Geunes, J., 2008. Predictive-reactive scheduling on a single resource with uncertain future jobs. *European Journal of Operational Research*, 189 (3), 1267–1283.

Zuo, X., Mo, H., and Wu, J., 2009. A robust scheduling method based on a multi-objective immune algorithm. *Information Sciences*, 179, 3359–3369.