ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-687

Schedule Generation Schemes for Job Shop Problems with Fuzziness

Juan José Palacios[1] and **Camino R. Vela**[2] and **Inés González-Rodríguez**[3] and **Jorge Puente**[4]

Abstract. We consider the job shop scheduling problem with fuzzy durations and expected makespan minimisation. We formally define the space of *semi-active* and *active fuzzy schedules* and propose and analyse different schedule-generation schemes (SGSs) in this fuzzy framework. In particular, we study dominance properties of the set of schedules obtained with each SGS. Finally, a computational study illustrates the great difference between the spaces of active and the semi-active fuzzy schedules, an analogous behaviour to that of the deterministic job shop.

1 Introduction

Scheduling is a research field of great importance, involving complex combinatorial constraint-satisfaction optimisation problems and with relevant applications in industry, finance, welfare, education, etc [13]. To enhance the applicability of scheduling, part of the research in this field has been devoted to modelling the uncertainty and vagueness pervading real-world situations, with great diversity of approaches [9]. In particular, fuzzy sets have been used in different manners, ranging from representing incomplete or vague states of information to using fuzzy priority rules with linguistic qualifiers or preference modelling [4]. They are also emerging as an interesting tool for improving solution robustness, a much-desired property in real-life applications [10, 15].

A key issue in scheduling is the definition of subsets of feasible solutions and the study of their properties, in particular, whether they are guaranteed to contain at least one optimal solution. For classical deterministic scheduling, the best known are the sets of semi-active, active and non-delay (or dense) schedules, and it is common practice to restrict the search to some of these subspaces. This is achieved using schedule generation schemes (SGSs) which, given an operation processing order, produce a schedule (an assignment of start times to all operations) based on this ordering. SGSs are extensively used in (meta)heuristic procedures and can also be viewed as branching schemes of exact search methods. It is essential to have proper SGSs, to know which is the set of schedules obtainable with a given SGS and how it relates with the schedule categories and to study the theoretical ability of any SGS to reach the optimum. Surprisingly enough, although we can find some ad-hoc extensions of deterministic SGSs to the fuzzy framework, no effort has been made to give precise definitions for types of schedules when fuzzy times are involved, nor have SGSs been defined and studied systematically in this framework.

[1] University of Oviedo, Spain, email: palaciosjuan@uniovi.es
[2] University of Oviedo, Spain, email: crvela@uniovi.es
[3] University of Cantabria, Spain, email: ines.gonzalez@unican.es
[4] University of Oviedo, Spain, email: puente@uniovi.es

In this paper, we intend to fill the existing gap in the literature. Inspired by the work of [1],[18],[19] for different deterministic scheduling problems, we provide a formal definition of the concepts of semi-active and active schedules as well as several SGSs for the fuzzy job shop problem with expected makespan minimisation (FJSP). We shall study the relationship between different types of schedules and the sets generated by SGSs, and investigate whether such sets necessarily contain one optimal schedule. Finally, we shall provide computational results to compare the different SGSs.

2 The Fuzzy Job Shop Problem

The *job shop scheduling problem*, or *JSP* in short, consists in scheduling a set of n jobs $J_1, \ldots, J_n$ to be processed on a set of m physical resources or machines $M_1, \ldots, M_m$. Each job J_j, $j = 1, \ldots, n$, consists of $m_j \leq m$ tasks or operations $(o(j,1), \ldots, o(j,m_j))$ to be sequentially scheduled (*precedence constraints*). Each task $o(j,l)$ needs the exclusive use of a machine $\mu_{o(j,l)}$ for its whole processing time $d_{o(j,l)} > 0$ (*capacity constraints*). There is no preemption, i.e. all operations must be processed without interruption and no reentrance, i.e., operations within a job are processed by different machines: $\forall j$, $\mu_{o(j,l)} \neq \mu_{o(j,l')}, \forall l \neq l'$. A solution to this problem is a *schedule*–an allocation of starting times for all operations – which is *feasible*, in the sense that all constraints hold, and is also optimal according to some criterion. Here, we consider the objective of minimising the makespan C_{max}, which is the time lag from the start of the first operation until the end of the last one. This is the most commonly considered regular (non-decreasing with task processing times) performance measure.

In order to simplify notation, we assume w.l.o.g. that tasks are indexed from 1 to $N = \sum_{j=1}^{n} m_j$, so we can refer to a task $o(j,l)$ by its index $o = \sum_{i=1}^{j-1} m_i + l$. The machine, duration, starting time and completion time of a task o are denoted respectively μ_o, d_o, S_o and C_o (notice the last two depend on the schedule). The set of tasks is denoted $O = \{0, 1, \ldots, N\}$, where 0 is an initial dummy operation, taken to be the first operation of each job (i.e. $o(j,0) = 0, \forall j = 1, \ldots, n$) and such that $d_0 = S_0 = 0$. Finally, a feasible schedule will be represented by the vector of operation starting times $\boldsymbol{t}$, where $t_o = S_o$ is the starting time of operation $o \in \{1, \ldots, N\}$ (in our case, a triangular fuzzy number, as described below).

2.1 Uncertain Durations as Fuzzy Numbers

In real-life applications, it is often the case that the exact time it takes to process a task is not known in advance and only some uncertain knowledge about the duration is available. The crudest representation

for uncertain processing times would be a human-originated confidence interval. If some values appear to be more plausible than others, a natural extension is a fuzzy interval or fuzzy number (cf. [5]). A *fuzzy interval* A is a fuzzy set on the reals with membership function $\mu_A : \mathbb{R} \to [0,1]$ such that its α-cuts $A_\alpha = \{r \in \mathbb{R} : \mu_A(r) \geq \alpha\}$, $\alpha \in (0,1]$, are intervals (bounded or not). The *support* of A is $A_0 = \{r \in \mathbb{R} : \mu_A(r) > 0\}$ and the *modal values* are those in A_1. A *fuzzy number* B is a fuzzy interval whose α-cuts are closed intervals, denoted $B_\alpha = [\underline{b}_\alpha, \overline{b}_\alpha]$, with compact support and unique modal value.

The simplest model of fuzzy interval is a *triangular fuzzy number* or *TFN*, using an interval $[a^1, a^3]$ of possible values and a modal value a^2 in it. A TFN A is denoted $A = (a^1, a^2, a^3)$ and its membership function takes the following triangular shape:

$$\mu_A(r) = \begin{cases} \frac{r-a^1}{a^2-a^1} & : a^1 \leq r \leq a^2 \\ \frac{r-a^3}{a^2-a^3} & : a^2 < r \leq a^3 \\ 0 & : r < a^1 \text{ or } a^3 < r \end{cases} \quad (1)$$

In the job shop, we essentially need two operations on fuzzy numbers, the sum and the maximum. In principle, these are obtained by extending the corresponding operations on real numbers using the *Extension Principle*. However, computing the sum or maximum of two fuzzy numbers is cumbersome if not intractable in general, because it requires evaluating two sums or two maxima for every value $\alpha \in [0,1]$. For the sake of simplicity and tractability of numerical calculations, we follow [6] and approximate the results of these operations by linear interpolation on the three defining points of each TFN (an approach also taken, for instance, in [3] or [11]). The approximated sum coincides with the actual sum, so for any pair of TFNs A and B:

$$A + B = (a^1 + b^1, a^2 + b^2, a^3 + b^3) \quad (2)$$

As for the maximum, for any two TFNs A, B, if $F = \max(A, B)$ denotes their maximum and $G = (\max\{a^1, b^1\}, \max\{a^2, b^2\}, \max\{a^3, b^3\})$ the approximated value, it holds that $\forall \alpha \in [0,1]$, $\underline{f}_\alpha \leq \underline{g}_\alpha, \overline{f}_\alpha \leq \overline{g}_\alpha$. The approximated maximum G is thus a TFN which artificially increases the value of the actual maximum F, maintaining the support and modal value, that is, $F_0 = G_0$ and $F_1 = G_1$. This approximation can be trivially extended to the case of more than two TFNs.

The membership function μ_A of a fuzzy number A can be interpreted as a possibility distribution on the reals; this allows to define the *expected value* of a fuzzy number [8], given for a TFN A by

$$E[A] = \frac{1}{4}(a^1 + 2a^2 + a^3). \quad (3)$$

The expected value coincides with the *neutral scalar substitute* of a fuzzy interval and can also be obtained as the centre of gravity of its *mean value* or using the *area compensation* method [4]. It induces a total ordering $\leq_E$ in the set of fuzzy intervals [6], where for any two fuzzy intervals A, B $A \leq_E B$ if and only if $E[A] \leq E[B]$. Clearly, for any two TFNs A and B, if $\forall i \in \{1,2,3\}, a^i \leq b^i$, then $A \leq_E B$.

2.2 Problem Statement

In analogy to the original problem, our objective is to find a fuzzy schedule with optimal makespan. However, neither the maximum nor its approximation define a total ordering in the set of TFNs. Using ideas similar to stochastic scheduling, we use the total ordering provided by the expected value, considering that the objective is to minimise the expected makespan $E[C_{max}]$. The resulting problem will be denoted $J|fuzzd_o|E[C_{max}]$ and can be formulated as follows:

$$\min E[C_{max}(S)] = E[\max_{1 \leq j \leq n} C_{o(j,m)}] \quad (4)$$

subject to:

$$\forall i \; C^i_o = S^i_o + d^i_o, \; \forall o \in O \quad (5)$$

$$\forall i \; S^i_{o(j,l)} \geq C^i_{o(j,l-1)}, \; 1 \leq l \leq m_j, 1 \leq j \leq n \quad (6)$$

$$\forall i \; S^i_o \geq C^i_{o'} \; \vee \; \forall i \; S^i_{o'} \geq C^i_o, \forall o \neq o' \in O : \mu_o = \mu_{o'} \quad (7)$$

Where $\forall i$ represents $\forall i \in \{1,2,3\}$.

Clearly, the FJSP is NP-hard, since setting all processing times to crisp figures yields the classical JSP.

Notice that the schedule is fuzzy in the sense that the starting, processing and completion times of each task are fuzzy numbers, seen as possibility distributions on the actual values they may take. However, there is no uncertainty regarding the order in which operations must be processed: once the starting times have been allocated, they establish clear orderings among operations in the same machine.

Algorithm 1 SGS Generic Algorithm

Require: an instance of $J|fuzzp_o|E[C_{max}]$, *P*, and a task order, π
Ensure: a schedule $\boldsymbol{t}$ for *P* according to π
1. $A = \{o(j,1) : 1 \leq j \leq n\}$
while A $\neq \emptyset$ **do**
2. compute the eligible set E $\subseteq$ A
3. select $o(j^*, l^*) = \arg\min\{\pi_{o(j,l)} : o(j,l) \in E\}$
4. $S_{o(j^*,l^*)} = ES_{o(j^*,l^*)}$
5. $A = A - \{o(j^*, l^*)\}(\cup\{o(j^*, l^* + 1)\}$ if $l^* < m_{j^*})$
end while
return $\boldsymbol{t}$, where $t_i = S_i, i = 1, \ldots, N$

3 Schedule Generation Schemes

A general framework for a SGS is provided in Algorithm 1: given a task order π (which can be interpreted as a priority vector), it allows to build different types of schedules, depending on the actual instantiation of some of its actions.

The generic algorithm builds the schedule in N iterations. At each iteration, the SGS computes a set of *eligible* tasks, *E*, which is a subset of the set of *available* tasks, *A*, containing the tasks that are candidates to be scheduled at the current iteration. In steps 3 and 4 the SGS selects the operation $o(j^*, l^*) \in E$ with the highest priority according to π and computes its *Earliest feasible Starting time* (ES) based on an *Appending* (ESA) or *Insertion* (ESI) strategy.

This framework covers a wide range of interesting SGSs, as we shall see in the sequel. However, it does not comprise all possible SGSs, in particular those where a non-available operation may be selected for scheduling or where starting times may be later modified in the schedule-building process.

3.1 Computing Earliest Feasible Starting Times

In the SGS generic algorithm, once a task has been selected, it is scheduled at its earliest feasible starting time ES. Depending on how this value is computed, we distinguish between *appending* SGS and *insertion* SGS.

In an appending scheme, an unscheduled task can be scheduled only after all tasks that have been previously scheduled in its machine

and its job. Formally, let $o(j,l)$ be the task for which the starting time must be computed, let $k = \mu_{o(j,l)}$ be the machine required by $o(j,l)$ and let $\lambda(k) \in O$ denote the latest task scheduled (at the current iteration) on machine k. Then, $ESA_{o(j,l)}$ can be computed in $\mathcal{O}(1)$ as follows:

$$ESA_{o(j,l)} = \max\{C_{\lambda(k)}, C_{o(j,l-1)}\} \tag{8}$$

In an insertion scheme, an unscheduled task $o(j,l)$ may be scheduled before tasks already scheduled on its machine, provided that the starting time of each of these tasks does not change. Hence, the scheme searches for the first insertion position where the selected task can fit without delaying the subsequent tasks already scheduled. Taking into account the definition of starting and completion times in the FJSP, the insertion position must fit "in each component" of the TFN. More precisely, let η_k be the number of tasks scheduled on machine k and let $\sigma_k = (0, \sigma(1,k), \ldots, \sigma(\eta_k, k))$, with $\sigma(\eta_k, k) = \lambda(k)$ denote the partial processing order of tasks already scheduled in machine k. If a position q, $0 \leq q < \eta_k$, is such that for all $i \in \{1,2,3\}$:

$$\max\{C^i_{\sigma(q,k)}, C^i_{o(j,l-1)}\} + d^i_{o(j,l)} \leq S^i_{\sigma(q+1,k)} \tag{9}$$

then q is a *feasible insertion position for operation* $o(j,l)$ between operations $\sigma(q,k)$ (possibly the dummy first task 0) and $\sigma(q+1,k)$. If there exists at least one position q verifying (9), we take $\overline{q} = \min_{q \text{ verifying (9)}} q$ and

$$ESI_{o(j,l)} = \max\{C^i_{\sigma(\overline{q},k)}, C^i_{o(j,l-1)}\} \tag{10}$$

Otherwise $ESI_{o(j,l)} = ESA_{o(j,l)}$

The earliest starting time of an eligible task in an insertion scheme can be computed in $\mathcal{O}(m)$, since there are at most $m-1$ tasks scheduled on machine $k = \mu_{o(j,l)}$

4 Schedule Categories and SGSs

The set Σ of feasible solutions usually constitutes a huge search space. Hence, it is common in deterministic scheduling to restrict the search to smaller subsets of Σ which define categories of schedules. Among these, the best known are the sets of semiactive, active and non-delay schedules [13]. A set of schedules of a given category is said to be *dominant* w.r.t. an objective function if it contains at least one optimal solution. In the following, we will always consider dominance w.r.t. expected makespan. A SGS is *complete* for a category if it can be used to generate all the schedules of this category.

4.1 Semi-active Schedules

For deterministic shop scheduling, the definition of semi-active schedules is based on the concept of *local left shift*, a change that consists in "moving an operation block to the left on the Gantt chart while preserving the operation sequences" [18]. This can be interpreted in the fuzzy case as follows.

Definition 1 *Let $\boldsymbol{t}$ be a feasible schedule, then a* local left shift *of a task o in $\boldsymbol{t}$ is a move giving another feasible schedule $\boldsymbol{s}$ where*

$$\begin{gathered}\exists i \in \{1,2,3\} : s^i_o = t^i_o - 1 \wedge \forall j \neq i \; s^j_o = t^j_o \\ s_{o'} = t_{o'} \forall o' \in O - \{o\}\end{gathered} \tag{11}$$

Definition 2 *A semi-active schedule is a feasible schedule in which none of the tasks can be locally left-shifted.*

Notice that for any feasible schedule that is not semi-active, there exists a sequence of local left shifts that produces a semi-active schedule without increasing any of the makespan components, C^i_{max}, and, therefore, without increasing the expected makespan. Hence, the set of semi-active schedules is strictly contained in the set of feasible schedules and is dominant for the FJSP with expected makespan minimization.

We are now in position of defining a SGS that produces semi-active schedules.

Definition 3 SemiActiveSGS *is an appending SGS where the eligible set E equals the set of available operations A, i.e., $E = A$.*

Theorem 1 SemiActiveSGS *generates only semi-active schedules and it is complete in this set.*

Sketch of Proof Schedules generated by SemiActiveSGS are always semi-active because every operation $o \in O$ ESA_o is assigned the least possible value, so it is unfeasible to reduce any of its components, and no local left-shift is available. On the other hand, given a semiactive schedule $\boldsymbol{t}$, we take π to be the topological order from the constraint graph that represents the precedence and capacity constraints between operations in $\boldsymbol{t}$ (this order always exists because, being $\boldsymbol{t}$ feasible, the graph is acyclic). For any operation ordering π, SemiActiveSGS(π) schedules all operations following exactly the same order π, so in particular SemiActiveSGS(π) $= \boldsymbol{t}$. □

Corollary 2 *The set of schedules generated by* SemiActiveSGS *is dominant.*

4.2 Active schedules

Given a feasible schedule $\boldsymbol{t}$ where no local left shifts are possible, a *global left shift* of an operation o is a move that allows "to start an operation earlier without delaying any other operation" [18]. More formally:

Definition 4 *Let $\boldsymbol{t}$ be a feasible schedule, then a* left shift *of an operation o in $\boldsymbol{t}$ is a move giving another feasible schedule $\boldsymbol{s}$ where:*

$$\begin{gathered}\exists i \in \{1,2,3\} : s^i_o < t^i_o \wedge \forall j \neq i \; s^j_o \leq t^j_o \\ s_{o'} = t_{o'} \forall o' \in O - \{o\}\end{gathered} \tag{12}$$

Definition 5 *Let $\boldsymbol{t}$ be a feasible schedule, then a* global left shift *of a task o in $\boldsymbol{t}$ is a left shift of o that is not obtainable by a sequence of local left shifts.*

Definition 6 *An active schedule is a feasible schedule where no global or local left shift lead to a feasible schedule.*

Notice that an active schedule contains no feasible insertion positions, because if an insertion position existed, this would allow for at least one global left shift. Also, given any semi-active but non-active schedule, it is always possible to perform a sequence of global left shift moves in order to build an active schedule without increasing any component of the starting times of tasks. Hence, the set of active schedules is a strict subset of the semi-active ones and remains dominant.

In the following we study different ways of generating active schedules, starting with a straightforward insertion version of the general SGS algorithm.

Definition 7 ActiveSGS *is an insertion SGS where the eligible set E is the whole set of available operations A, i.e., $E = A$.*

Theorem 3 ActiveSGS *generates only active schedules and it is complete in this set.*

Proof Let π be a task processing order, let $\boldsymbol{t} = \text{ActiveSGS}(\pi)$ and let $\sigma_k = (0, \sigma(1,k), \ldots, \sigma(\eta_k, k))$ denote the partial sequencing order in which operations are scheduled on a machine k according to $\boldsymbol{t}$. If $\boldsymbol{t}$ is not active, there must exist a task $o(j,l)$ scheduled in its machine k at a position $p_k \in \{2, \ldots, \eta_k\}$ such that for $o(j,l)$ there exists a feasible insertion position $q < p_k$ in σ_k. Thus, there exists a feasible schedule $\boldsymbol{s}$ such that $s_{o'} = t_{o'}, \forall o' \neq o(j,l)$ and

$$\forall i\ s^i_{o(j,l)} + d^i_{o(j,l)} \leq \min\{t^i_{o(j,l+1)}, t^i_{\sigma(q+1,k)}\},$$
$$\forall i\ s^i_{o(j,l)} = \max\{C^i_{o(j,l-1)}, C^i_{\sigma(q,k)}\} < t^i_{o(j,l)}.$$

But this is absurd because if such feasible insertion position exists at the end of the algorithm, it must also exist when operation $o(j,l)$ is to be scheduled by ActiveSGS and, in this case, $t^i_{o(j,l)} = ESI^i_{o(j,l)}$ can never be greater than $s^i_{o(j,l)}$ for any component i.

Conversely, let $\boldsymbol{t}$ be an active schedule and let π be the task processing order obtained as the topological order of the constraint graph representing $\boldsymbol{t}$. Since it is active, no feasible insertion positions can exist in $\boldsymbol{t}$. Therefore, $\text{ActiveSGS}(\pi)$ will schedule every task $o(j,l)$ with starting time $ESI_{o(j,l)} = ESA_{o(j,l)} = \max\{C_{\lambda(k)}, C_{o(j,l-1)}\}$ where $\lambda(k)$ is the operation preceding $o(j,l)$ in its machine k according to π, i.e. $ESI_{o(j,l)} = t_{o(j,l)}$. It thus follows that $\boldsymbol{t} = \text{ActiveSGS}(\pi) = \text{SemiActiveSGS}(\pi)$ □

Corollary 4 *The set of schedules generated by* ActiveSGS *is dominant.*

4.2.1 *The* fG&T-SGS *algorithms*

The Giffler-Thompson Algorithm or G&T in short ([7]) is probably the most famous active schedule generation scheme for deterministic job shop problem, having been used in a variety of settings. It is an appending algorithm where, given the task o^* with earliest possible completion time C^* at the current step, the set E of eligible operations (also referred to as *conflict set*) is the set of operations processed in the same machine as o^* which may start before C^*.

G&T provides a complete and constructive heuristic method to search for solutions in search spaces of reasonable size and has been used as a branching schema for the deterministic JSP in exact methods, such as branch and bound [2] or best-first search [17]. Also, G&T allows further reductions of the search space by including a parameter that bounds the length of time that a machine is allowed to remain idle on the chance that a "more critical" job will soon become available [19].

We can find some ad-hoc extensions of G&T in the fuzzy scheduling literature, from the earliest one in [16] to the most recent one in [12]. The variety of existing proposals illustrates that extending G&T is far from trivial. The first difficulty appears when computing the earliest completion time C^* at each current step. If it is computed as the minimum completion time of all the unscheduled tasks currently available, it may not correspond to the completion time of any specific task because a set of TFNs is not closed under the minimum. In consequence, it may not make sense to consider only one machine when computing the eligible set.

A possible solution is to build the eligible set E with all tasks o that "can start before C^*", which in fuzzy framework means that $\exists i ESA^i_o < (C^*)^i$, since C^* is previous to ESA_o only if $\forall i, (C^*)^i \leq ESA^i_o$. This is the basis for the first SGS extending G&T:

Definition 8 *The* fG&T-SGS1 *algorithm is an appending SGS where the eligible set E is computed as follows:*

$$\begin{aligned} C^* &= \min\{ESA_o + d_o : o \in A\} \\ E &= \{o \in A : \exists i\ ESA^i_o < (C^*)^i\} \end{aligned} \tag{13}$$

Theorem 5 fG&T-SGS1 *generates only active schedules, but it is not complete in this set and it is not dominant.*

Sketch of Proof We first prove by contradiction that fG&T-SGS1 generates active schedules. Let π be a task processing order and let us suppose that $\boldsymbol{t} = \text{fG\&T-SGS1}(\pi)$ is not active. Let $\sigma_k = (0, \sigma(1,k), \ldots, \sigma(\eta_k, k))$ denote the partial sequencing order in which operations are scheduled on a machine k according to $\boldsymbol{t}$. Reasoning as in Theorem 3, there must exist a task $o(j,l)$ scheduled in its machine k at a position $p_k \in \{2, \ldots, \eta_k\}$, a feasible schedule $\boldsymbol{s}$ with $s_{o'} = t_{o'}, \forall o' \neq o(j,l)$ and a position $q < p_k$ such that

$$\forall i\ s^i_{o(j,l)} + d^i_{o(j,l)} \leq \min\{t^i_{o(j,l+1)}, t^i_{\sigma(q+1,k)}\},$$
$$\forall i\ s^i_{o(j,l)} = \max\{C^i_{o(j,l-1)}, C^i_{\sigma(q,k)}\} < t^i_{o(j,l)}.$$

For the feasible position q to exist in $\boldsymbol{t}$, it must be the case that fG&T-SGS1 has scheduled operation $\sigma(q+1,k)$ before $o(j,l)$.

Also, $o(j,l)$ cannot have been in the set A when $\sigma(q+1,k)$ was to be scheduled. This is proved by contradiction using the fact that q is a feasible insertion position.

A direct consequence is that $o(j,l-1)$ cannot have been scheduled either. In fact, $o(j,l-1)$ cannot even have been in A when $\sigma(q+1,k)$ was to be scheduled. This is again proved by contradiction, using the fact that $\boldsymbol{s}$ and $\boldsymbol{t}$ are identical for every operation other than $o(j,l)$ and that a feasible insertion position exists.

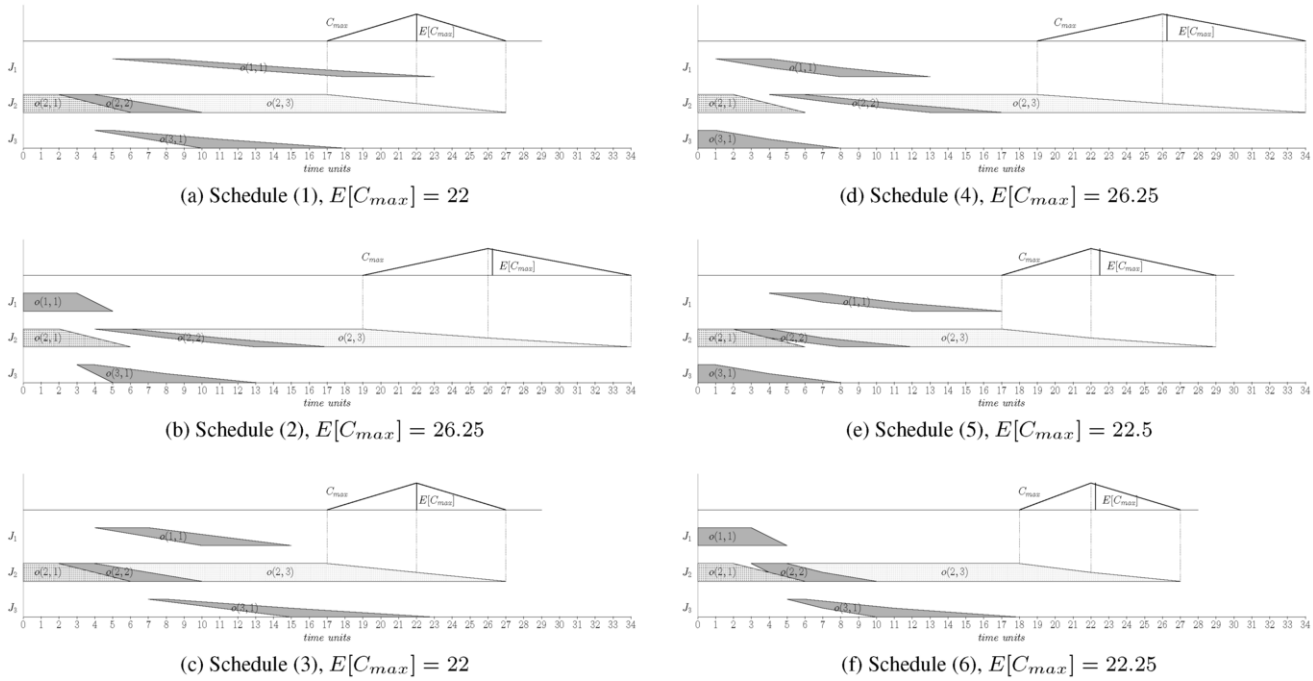
By repeating this argument "backwards" for all operations preceding $o(j,l)$ in its job, we conclude that $o(j,1)$ cannot have been in A when $\sigma(q+1,k)$ was scheduled, which is clearly absurd because A is initialised with the first task of every job.

To show that fG&T-SGS1 does not generate all active schedules nor is it complete, consider a problem with 3 jobs and 3 machines where durations are $d_{o(1,1)} = (3,4,5)$, $d_{o(2,1)} = (2,4,6)$, $d_{o(2,2)} = (2,3,4)$, $d_{o(2,3)} = (13,15,17)$, $d_{o(3,1)} = (1,4,8)$ and with the following machine requirements $\mu_{o(1,1)} = 1$, $\mu_{o(2,1)} = 2$, $\mu_{o(2,2)} = 1$, $\mu_{o(2,3)} = 3$, $\mu_{o(3,1)} = 1$. Figure 1 shows the job-oriented Gantt charts adapted to TFNs (following [6]) of all six feasible active schedules, including the two optimal solutions with $C_{max} = (17,22,27)$ (solutions (1) and (3)). In this case, it is easy to see that fG&T-SGS1 cannot generate any of the optimal (active) solutions. □

The incompleteness of fG&T-SGS1 stems from the fact that a set of TFNs is not closed under the minimum, i.e., C^* may not correspond to the earliest completion time of an operation in A; we can only guarantee that $(C^*)^i$ does correspond to the i-th component of the earliest completion time of an operation in A. Taking this into account, we propose an alternative extension of G&T.

Definition 9 *The* fG&T-SGS2 *algorithm is an appending SGS where the eligible set E is computed as follows.*

$$\begin{aligned} C^* &= \min\{ESA_o + d_o : o \in A\} \\ A^* &= \{o \in A : \exists i\ ESA^i_o + d^i_o = (C^*)^i\} \\ E &= \{o \in A : \forall o' \in A^* \exists i\ ESA^i_o < ESA^i_{o'} + d^i_{o'}\} \end{aligned} \tag{14}$$



(a) Schedule (1), $E[C_{max}] = 22$

(b) Schedule (2), $E[C_{max}] = 26.25$

(c) Schedule (3), $E[C_{max}] = 22$

(d) Schedule (4), $E[C_{max}] = 26.25$

(e) Schedule (5), $E[C_{max}] = 22.5$

(f) Schedule (6), $E[C_{max}] = 22.25$

Figure 1: Gantt chart of the schedules of all active solutions for the example in Theorem 5.

Theorem 6 fG&T-SGS2 *generates only active schedules and it is complete in this set.*

Sketch of Proof The argument that fG&T-SGS2 generates active schedules is analogous to that given for fG&T-SGS1 in Theorem 5.

To see that fG&T-SGS2 is complete, let $\boldsymbol{t}$ be an active schedule, let σ be the task processing order obtained from the topological ordering of the constraint graph represented by $\boldsymbol{t}$ and let σ_k be the partial order determined by σ for a particular machine k. We prove that for an operation processing order π containing all partial orders represented by σ_k and $\boldsymbol{s} = \text{fG\&T-SGS2}(\pi)$, we have $\boldsymbol{s} = \boldsymbol{t}$. It suffices to show that, if σ' is the task processing order obtained from the topological ordering of the constraint graph represented by $\boldsymbol{s}$, $\forall k$ $\sigma_k = \sigma'_k$.

Let us suppose that there exists at least one k such that $\sigma_k \neq \sigma'_k$ and let $a = o(j,l) = \sigma(q,k)$ be the first operation in σ that is scheduled in its machine k in a different order from σ. This means that there exists an operation requiring the same machine as a, $b = \sigma(q',k)$, $q' > q$, that will be scheduled by fG&T-SGS2 before a. Notice that, $b \in E$ and $a \notin E$. Also, without loss of generality, we may assume that $a \in A$. Finally, notice that, being an active schedule, in $\boldsymbol{t}$ there are no feasible insertion positions, that is, $\exists i\ ESA^i_a < ESA^i_b + d^i_b$.

If $b \in A^*$, since $a \in A - E$, there must exist at least one operation $o \in A^* \subseteq E$ such that $\forall i\ ESA^i_o + d^i_o \leq ESA^i_a$. o cannot share job with a or b. If it requires a machine $k' \neq k$, it can be scheduled before b without any change in any of the partial orders in σ. Using this argument a finite number of times, eventually $\forall x \in A^*, \mu_x = k$. This, together with the fact that $\boldsymbol{t}$ is active, leads to having $a \in E$, which is a contradiction. If $b \notin A^*$, $b \in E$ means that $\forall o \in A^*$ $\exists i : ESA^i_b < ESA^i_o + d^i_o$. Reasoning analogously to the case when $b \in A^*$, we conclude that it is impossible to schedule b before a, which is a contradiction. □

Corollary 7 *The set of schedules generated by* fG&T-SGS2 *is dominant.*

5 Empirical Behaviour

Having studied the different features of each proposed SGS, in this section we intend to illustrate their behaviour in practice. To this end, we will analyse the quality of the solutions generated by each SGS from a broad sample of operation processing orders, which will also offer a picture of the different schedule spaces. This study is carried out on the fuzzy instances from [14], a set of 12 fuzzified versions of what are considered to be some of the hardest instances for the JSP. For each instance, we generate $T = 1000$ random feasible task orderings and evaluate each ordering using the four SGSs proposed in this paper.

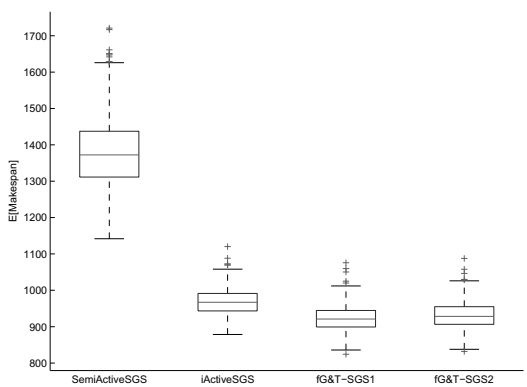


Figure 2: $E[C_{max}]$ for 1000 task orderings for instance ABZ9.

The box-plot in Figure 2 corresponds to the expected makespan obtained with the T task orderings using the different SGSs. It corresponds to instance ABZ9; for the remaining instances, the behaviour is very similar. As expected, the semi-active solutions generated by

SemiActiveSGS are much worse than the active ones obtained with the other SGSs; this is due to the size and features of the related space of solutions. These results confirm the clear difference, also in the fuzzy framework, between the spaces of active schedules and semi-active ones. Differences between active SGSs are on the other hand not so clear, even if ActiveSGS seems to yield slightly worse solutions than the two extensions of G&T.

A better assessment of the SGSs is achieved through a series of non-parametric statistical inference tests, having rejected the hypotheses of normality for all instances with preliminary Kolmogorov-Smirnov tests. For each instance we run a Friedman two-way analysis of variance by ranks. As for the box-plots, results are very similar for all instances, and show that there is a significant difference between the samples corresponding to each SGS. According to the mean ranks provided by the test on ABZ9, the SGS can be ranked according to the average quality of the solutions as follows: the best one would be fG&T-SGS1 (1.4215), followed by fG&T-SGS2 (1.7565), then ActiveSGS (2.822) and finally SemiActiveSGS (4); the results for the remaining instances are very similar. Additionally, a Mann-Whitney U test is run on each pair of samples. According to this test, for instances FT10, FT20 and LA25, there are not significant differences between fG&T-SGS1 and fG&T-SGS2 (with p-values 0.288, 0.206 and 0.129 respectively). For the remainng instances, a p-value< 0.01 indicates that there are significant differences between both extensions of G&T.

An explanation for these results is that fG&T-SGS1 maps the processing orders to a subspace of the active schedules with good solutions in average, even if it is not guaranteed to contain any optimal solution. For large instances with a huge solution space, this reduction may prove worthwhile. However, for small instances (or if the SGS is to be used in a exact algorithm) it may be better to use fG&T-SGS2 or ActiveSGS, which allow to search across the whole space of active schedules. In fact, although both are complete, the mapping defined by fG&T-SGS2 seems significantly better in average quality.

The behaviour shown for the fuzzy setting is consistent with the deterministic JSP, where active schedules are good in average (and much better than semi-active ones) and form a dominant set. Also, in the crisp case the G&T algorithm can be modified in order to further reduce the search space; at the extreme, the search space is constrained to that of non-delay schedules, where a machine cannot be idle if there is an operation that can be executed in it. Experience demonstrates that the mean value of solutions tends to improve with the reduction of the search space, despite the risk of losing the optimal solution.

6 Conclusions

This papers provides the first formal definition and study of types of feasible fuzzy schedules and related schedule generation schemes for the job shop problem with fuzzy processing times. We have shown that dominance and completeness are lost when considering a simple extension of the G&T algorithm, while an insertion SGS algorithm and a more sophisticated extension of the G&T are both complete and dominant. Additional experimental results have confirmed the differences between semi-active and active subspaces and shown that narrowing the search space can improve the average quality of schedules even if dominance is lost. We believe both the theoretical and experimental results can provide a guide for designing SGS and incorporate them both into metaheuristic and exact search methods.

As future work, we plan to extend this study to smaller sets of schedules, such as non-delay. Also, the fuzzy setting allows for alternative definitions of left shifts and, consequently, (semi)active schedules, thus admiting more constraints in the solution space than those existing in the deterministic job shop which may be worth exploring.

ACKNOWLEDGEMENTS

This research has been supported by the Spanish Government under research grants FEDER TIN2010-20976-C02-02 and MTM2010-16051 and by the Principality of Asturias (Spain) under grants Severo Ochoa BP13106 and FC-13-COF13-035.

REFERENCES

[1] C. Artigues, P. Lopez, and P.D. Ayache, 'Schedule generation schemes for the job shop problem with sequence-dependent setup times: Dominance properties and computational analysis', *Annals of Operations Research*, **138**, 21–52, (2005).

[2] P. Brucker, B. Jurisch, and B. Sievers, 'A branch and bound algorithm for the job-shop scheduling problem', *Discrete Applied Mathematics*, **49**, 107–127, (1994).

[3] S-M. Chen and T-H. Chang, 'Finding multiple possible critical paths using fuzzy PERT', *IEEE Transactions on Systems, Man, and Cybernetics–Part B:*, **31**(6), 930–937, (2001).

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

[5] D. Dubois and H. Prade, *Possibility Theory: An Approach to Computerized Processing of Uncertainty*, Plenum Press, New York (USA), 1986.

[6] P. Fortemps, 'Jobshop scheduling with imprecise durations: a fuzzy approach', *IEEE Transactions of Fuzzy Systems*, **7**, 557–569, (1997).

[7] B. Giffler and G. L. Thompson, 'Algorithms for solving production scheduling problems', *Operations Research*, **8**, 487–503, (1960).

[8] S. Heilpern, 'The expected value of a fuzzy number', *Fuzzy Sets and Systems*, **47**, 81–86, (1992).

[9] W. Herroelen and R. Leus, 'Project scheduling under uncertainty: Survey and research potentials', *European Journal of Operational Research*, **165**, 289–306, (2005).

[10] A. Kasperski and M. Kule, 'Choosing robust solutions in discrete optimization problems with fuzzy costs', *Fuzzy Sets and Systems*, **160**, 667–682, (2009).

[11] Q. Niu, B. Jiao, and X. Gu, 'Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time', *Applied Mathematics and Computation*, **205**, 148–158, (2008).

[12] J. J. Palacios, I. González-Rodríguez, C. R. Vela, and J. Puente, 'Swarm lexicographic goal programming for fuzzy open shop scheduling', *Journal of Intelligent Manufacturing*, (2013) *In press*.

[13] M. L. Pinedo, *Scheduling. Theory, Algorithms, and Systems.*, Springer, third edn., 2008.

[14] J. Puente, C. R. Vela, and I. González-Rodríguez, 'Fast local search for fuzzy job shop scheduling', in *Proceedings of ECAI 2010*, pp. 739–744. IOS Press, (2010).

[15] S. J. Sadjadi, R. Pourmoayed, and M.B. Aryanezhad, 'A robust critical path in an environment with hybrid uncertainty', *Applied Soft Computing*, **12**(3), 1087–1100, (2012).

[16] M. Sakawa and T. Mori, 'An efficient genetic algorithm for job-shop scheduling problems with fuzzy processing time and fuzzy duedate', *Computers & Industrial Engineering*, **36**, 325–341, (1999).

[17] M. Sierra and R. Varela, 'Pruning by dominance in best-first search for the job shop scheduling problem with total flow time.', *Journal of Intelligent Manufacturing*, **21**(1), 111–119, (2010).

[18] A. Sprecher, R. Kolisch, and A. Drexl, 'Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem', *European Journal of Operational Research*, **80**, 94–102, (1995).

[19] R. H. Storer, S. D. Wu, and R. Vaccari, 'New search spaces for sequencing problems with application to job shop scheduling', *Management Science*, **38**(10), 1495–1509, (1992).

[20] J. Wang, 'A fuzzy robust scheduling approach for product development projects', *European Journal of Operational Research*, **152**, 180–194, (2004).