A New Local Search for the Job Shop Problem with Uncertain Durations

Inés González-Rodríguez
Dept. of Mathematics, Statistics and Computing, University of Cantabria, 39005 Santander (Spain)
e-mail: ines.gonzalez@unican.es

Camino R. Vela and **Jorge Puente** and **Ramiro Varela**
Dept. of Computer Science and A.I. Centre, University of Oviedo, 33271 Gijón (Spain)
e-mail: {crvela,puente,ramiro}@uniovi.es

Abstract

In the sequel we consider the job shop scheduling problem with uncertain durations represented as triangular fuzzy numbers. We propose a new neighbourhood structure for local search, based on a new definition of critical path for fuzzy durations. A theoretical analysis of the proposed structure shows that it improves a previous one from the literature. It also shows that feasibility and connectivity hold, these being two highly desirable properties. Experimental results are reported which further illustrate the potential of the proposal.

Introduction

Scheduling problems form an important body of research since the late fifties, with multiple applications in industry, finance and science (Brucker & Knust 2006). Part of this research is devoted to fuzzy scheduling, in an attempt to model the uncertainty and vagueness pervading real-world situations. The approaches are diverse, ranging from representing incomplete or vague states of information to using fuzzy priority rules with linguistic qualifiers or preference modelling (Dubois, Fargier, & Fortemps 2003),(Słowiński & Hapke 2000).

Incorporating uncertainty to scheduling usually requires a significant reformulation of the problem and solving methods, in order that the problem can be precisely stated and solved efficiently and effectively. For instance, extending critical path analysis to ill-known processing times is far from being trivial (Dubois, Fargier, & Fortemps 2003). Furthermore, in classical scheduling the complexity of problems such as shop problems means that practical approaches to solving them usually involve heuristic strategies: simulated annealing, genetic algorithms, local search, etc (Brucker & Knust 2006). Some attempts have been made to extend these heuristic methods to fuzzy scheduling problems where uncertain durations are modelled via fuzzy intervals, most commonly and successfully for the flow shop problem: among others, a genetic algorithm is used in (Celano, Costa, & Fichera 2003) and a genetic algorithm is hybridised with a local search procedure in (Ishibuchi & Murata 1998). For the job shop, single objective problems are tackled using simulated annealing in (Fortemps 1997) and a genetic algorithm hybridised with local search in (González Rodríguez, Vela, & Puente 2007), whilst multiobjective problems are solved using genetic algorithms in (Sakawa & Kubota 2000) or (González Rodríguez *et al.* 2008) and using a neural approach in (Tavakkoli-Moghaddam, Safei, & Kah 2008).

In the following, we consider a job shop problem with task durations modelled as triangular fuzzy numbers. We propose new definitions of critical path and neighbourhood structure for local search. We give theoretical results which demonstrate the potential of the new proposal, further illustrated by the experimental results.

Uncertain Durations

In real-life applications, it is often the case that the exact duration of a task, i.e. the time it takes to be processed, is not known in advance. However, based on previous experience, an expert may have some knowledge (albeit uncertain) about the duration. In the literature, it is common to use fuzzy intervals to represent such uncertainty, as an alternative to probability distributions, which require a deeper knowledge of the problem and usually yield a complex calculus.

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

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

Triangular fuzzy numbers and more generally fuzzy intervals have been extensively studied in the literature (cf. (Dubois & Prade 1986)). A *fuzzy interval* Q is a fuzzy quantity (a fuzzy set on the reals) whose α-cuts $Q_\alpha = \{r \in \mathbb{R} : \mu_Q(r) \geq \alpha\}$, $\alpha \in (0, 1]$, are intervals (bounded or not). The *support* of Q is $Q_0 = \{r \in \mathbb{R} : \mu_Q(r) > 0\}$. A *fuzzy*

number is a fuzzy quantity whose α-cuts are closed intervals, with compact support and unique modal value.

In the job shop, we essentially need two operations on fuzzy quantities, the sum and the maximum. These are obtained by extending the corresponding operations on real numbers using the *Extension Principle*. However, computing the resulting expression is cumbersome, if not intractable. For the sake of simplicity and tractability of numerical calculations, we follow (Fortemps 1997) and approximate the results of these operations by a TFN, evaluating only the operation on the three defining points of each TFN. The approximated sum coincides with the sum of TFNs as defined by the Extension Principle, so for any pair of TFNs M and N:

$$M + N = (m^1 + n^1, m^2 + n^2, m^3 + n^3) \tag{2}$$

The same is not always true for the maximum $\vee$, although it is possible to prove that for any two TFNs M, N, if $F = N \vee M$ denotes their maximum and $G = (m^1 \vee n^1, m^2 \vee n^2, m^3 \vee n^3)$ its approximated value, it holds that:

$$\forall \alpha \in [0,1], \quad \underline{f}_\alpha \leq \underline{g}_\alpha, \overline{f}_\alpha \leq \overline{g}_\alpha. \tag{3}$$

where $[\underline{f}_\alpha, \overline{f}_\alpha]$ is the α-cut of F. In particular, F and G have identical 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_Q of a fuzzy quantity Q can be interpreted as a possibility distribution on the real numbers, so we naturally obtain the *possibility* Π and *necessity measure* N that $Q \leq r$, where r is a real number (Dubois & Prade 1986). These measures are used to define the *expected value* of a fuzzy quantity (Liu & Liu 2002), given for a TFN A by

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

The expected value coincides with the 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 (Dubois, Fargier, & Fortemps 2003). It induces a total ordering $\leq_E$ in the set of fuzzy intervals (Fortemps 1997), where for any two fuzzy intervals M, N $M \leq_E N$ if and only if $E[M] \leq E[N]$. Clearly, for any two TFNs A and B, if $\forall i, a^i \leq b^i$, then $A \leq_E B$.

The Job Shop Scheduling Problem

The classical *job shop scheduling problem*, also denoted *JSP*, consists in scheduling a set of jobs $\{J_1, \ldots, J_n\}$ on a set $\{M_1, \ldots, M_m\}$ of physical resources or machines, subject to a set of constraints. There are *precedence constraints*, so each job J_i, $i = 1, \ldots, n$, consists of m tasks $\{\theta_{i1}, \ldots, \theta_{im}\}$ to be sequentially scheduled. Also, there are *capacity constraints*, whereby each task θ_{ij} requires the uninterrupted and exclusive use of one of the machines for its whole processing time. A solution to this problem is a schedule, i.e. an allocation of starting times for each task, which, besides being *feasible* (i.e. all constraints hold), is *optimal* according to some criterion, most commonly that the *makespan* is minimal.

Expected Makespan Model

For a job shop problem instance of size $n \times m$ (n jobs and m machines), let $\boldsymbol{p}$ be a duration matrix and let $\boldsymbol{\nu}$ be a machine matrix such that p_{ij} is the processing time of task θ_{ij} and ν_{ij} is the machine required by θ_{ij}, $i = 1, \ldots, n$, $j = 1, \ldots, m$. Let σ be a feasible task processing order, i.e., a lineal ordering of tasks which is compatible with a processing order of tasks that may be carried out so that all constraints hold. A feasible schedule may be derived from σ using a *semi-active schedule builder*. Let $C_i(\sigma, \boldsymbol{p}, \boldsymbol{\nu})$ denote the completion time of job J_i and let $S_{ij}(\sigma, \boldsymbol{p}, \boldsymbol{\nu})$ and $C_{ij}(\sigma, \boldsymbol{p}, \boldsymbol{\nu})$ denote the starting and completion times of task θ_{ij}, $i = 1, \ldots, n$ $j = 1, \ldots, m$. These times are obtained as follows:

$$C_i(\sigma, \boldsymbol{p}, \boldsymbol{\nu}) = C_{im}(\sigma, \boldsymbol{p}, \boldsymbol{\nu}) \tag{5}$$

$$S_{ij}(\sigma, \boldsymbol{p}, \boldsymbol{\nu}) = C_{i(j-1)}(\sigma, \boldsymbol{p}, \boldsymbol{\nu}) \vee C_{rs}(\sigma, \boldsymbol{p}, \boldsymbol{\nu}) \tag{6}$$

$$C_{ij}(\sigma, \boldsymbol{p}, \boldsymbol{\nu}) = S_{ij}(\sigma, \boldsymbol{p}, \boldsymbol{\nu}) + p_{ij} \tag{7}$$

where θ_{rs} is the task preceding θ_{ij} in the machine according to the processing order σ, $C_{i0}(\sigma, \boldsymbol{p}, \boldsymbol{\nu})$ is assumed to be zero and, analogously, $C_{rs}(\sigma, \boldsymbol{p}, \boldsymbol{\nu})$ is taken to be zero if θ_{ij} is the first task to be processed in the corresponding machine. The *makespan* $C_{max}(\sigma, \boldsymbol{p}, \boldsymbol{\nu})$ is the maximum completion time of jobs $J_1, \ldots, J_n$:

$$C_{max}(\sigma, \boldsymbol{p}, \boldsymbol{\nu}) = \vee_{1 \leq i \leq n} (C_i(\sigma, \boldsymbol{p}, \boldsymbol{\nu})) \tag{8}$$

For the sake of a simpler notation, we may write, for instance, $C_{max}(\sigma)$ when the problem (hence $\boldsymbol{p}$ and $\boldsymbol{\nu}$) is fixed or even C_{max} when no confusion is possible regarding the task processing order.

If task processing times are fuzzy intervals, then the addition and maximum in the equations above are taken to be the corresponding operations on fuzzy intervals, approximated for the particular case of TFNs as proposed above. The obtained schedule will be a fuzzy schedule, where the starting and completion times of all tasks and the makespan are fuzzy intervals. Such intervals may be seen as possibility distributions on the values that the times may take. However, the task processing ordering σ that determines the schedule is crisp; there is no uncertainty regarding the order in which tasks are to be processed. In other words, we obtain a fuzzy schedule from a crisp task ordering.

To illustrate the schedule builder, consider a problem of 3 jobs and 2 machines with the following matrices for fuzzy processing times and machine allocation:

$$\boldsymbol{p} = \begin{pmatrix} (3,4,7) & (1,2,3) \\ (4,5,6) & (2,3,4) \\ (1,2,6) & (1,2,4) \end{pmatrix} \boldsymbol{\nu} = \begin{pmatrix} 1 & 2 \\ 2 & 1 \\ 2 & 1 \end{pmatrix}$$

For the feasible task order $\theta_{11}, \theta_{21}, \theta_{31}, \theta_{22}, \theta_{12}, \theta_{32}$, for instance, the starting and completion times for task θ_{22} will be $S_{22} = C_{21} \vee C_{11} = (4,5,7)$ and $C_{22} = S_{22} + p_{22} = (6,8,11)$. Figure 1 corresponds to the Gantt chart (adapted to TFNs following (Fortemps 1997)) of the resulting schedule, using different textures for tasks in different jobs. It shows the final makespan as well as the starting times of tasks scheduled in each machine: for M_1, $S_{11} = (0,0,0)$,

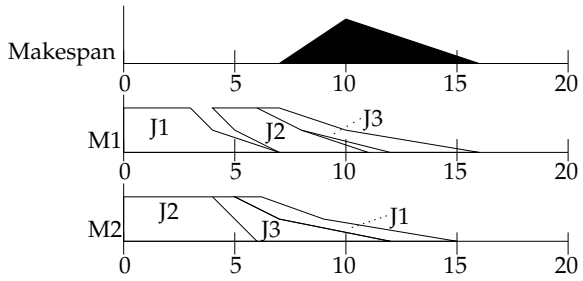

Figure 1: Gantt chart of the schedule represented by $\theta_{11}, \theta_{21}, \theta_{31}, \theta_{22}, \theta_{12}, \theta_{32}$

$S_{22} = (4, 5, 7)$, $S_{32} = (6, 8, 12)$, and for M_2, $S_{21} = (0, 0, 0)$, $S_{31} = (4, 5, 6)$, $S_{12} = (5, 7, 12)$.

Since we may build a feasible schedule from a feasible task processing order, we restate the goal of the job shop problem as that of finding an optimal task processing order, in the sense that makespan for the derived schedule is optimal. However, it is not trivial to optimise a fuzzy makespan, since neither the maximum nor its approximation define a total ordering in the set of TFNs. In the literature, this problem is tackled using ranking methods, lexicographical orderings, comparisons based on α-cuts or defuzzification methods. Here we use the concept of expected value for a fuzzy quantity and the total ordering it provides, and consider that the objective is to minimise the expected makespan $E[C_{max}(\sigma, \boldsymbol{p}, \boldsymbol{\nu})]$, a crisp objective function. An optimal task processing order is then defined as follows:

Definition 1. *Let Σ be the set of feasible task processing orders. A task processing order $\sigma_0 \in \Sigma$ is said to be* optimal *if and only if $E[C_{max}(\sigma_0)] = \min\{E[C_{max}(\sigma)] : \sigma \in \Sigma\}$*

A feasible task processing order may be represented by a decision variable $\boldsymbol{z} = (z_1, \ldots, z_{nm})$, where $1 \leq z_l \leq n$ for $l = 1, \ldots, nm$ and $|\{z_l : z_l = i\}| = m$ for $i = 1, \ldots, n$. This is a permutation with repetition (Bierwirth 1995), a permutation of the set of tasks, where each task is labelled with its job name, so the order of precedence among tasks requiring the same machine is given by the order in which they appear in the decision variable $\boldsymbol{z}$. For instance, the processing order from the example in Figure 1 is represented by the decision variable (1 2 3 2 1 3). Using this representation, the *expected makespan model* for the job shop (González Rodríguez, Vela, & Puente 2007) is expressed as follows:

$$\begin{cases} \min & E[C_{max}(\boldsymbol{z}, \boldsymbol{p}, \boldsymbol{\nu})] \\ \text{subject to:} & \\ & 1 \leq z_l \leq n, \quad l = 1, \ldots, nm, \\ & |\{z_l : z_l = i\}| = m, \quad i = 1, \ldots, n, \\ & z_l \in \mathbb{Z}^+, \quad l = 1, \ldots, nm. \end{cases} \tag{9}$$

The Disjunctive Graph Model Representation

A job shop problem instance may be represented by a directed graph $G = (V, A \cup D)$. Each node in the set V represents a task of the problem, with the exception of the dummy nodes *start* or 0 and *end* or $nm + 1$, representing tasks with null processing times. Task θ_{ij}, $1 \leq i \leq n$, $1 \leq j \leq m$, is represented by node $x = m(i-1) + j$. Arcs in A are called *conjunctive arcs* and represent precedence constraints (including arcs from node *start* to the first task of each job and arcs form the last task of each job to the node *end*). Arcs in D are called *disjunctive arcs* and represent capacity constraints. Set D is partitioned into subsets D_i, $D = \cup_{i=1,\ldots,m} D_i$, where D_i corresponds to machine M_i and includes an arc for each pair of tasks requiring that machine. Each arc is weighted with the processing time of the task at the source node (a TFN in our case). A feasible schedule task processing order σ is represented by an acyclic subgraph $G(\sigma)$ of G, $G(\sigma) = (V, A \cup R(\sigma))$, where $R(\sigma) = \cup_{i=1\ldots m} R_i(\sigma)$, $R_i(\sigma)$ being a hamiltonian selection of D_i. Figure 2 shows the solution graph for the example in Figure 1.

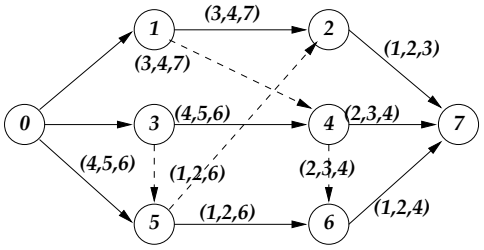

Figure 2: Solution graph $G(\sigma)$ for the processing order σ =(1 3 5 4 2 6) corresponding to the Gantt chart of Figure 1. $C_{max}(\sigma) = (7, 10, 16)$

In the crisp case, a *critical path* is defined as the longest path from node *start* to node *end* and a *critical arc* or *critical activity* is an arc or activity in a critical path. It is not trivial to extend these concepts and related algorithms to the problem with uncertain durations (cf. (Dubois, Fargier, & Fortemps 2003)). For the fuzzy job shop considered herein it may even be the case that the makespan (a TFN) does not coincide with the completion time of one job (unlike the crisp case).

Local Search

Part of the interest of critical paths stems from the fact that they may be used to define neighbourhood structures for local search. Roughly speaking, a typical local search schema starts from a given processing order, calculates its neighbourhood and then neighbours are evaluated (for instance, using the aforementioned semiactive schedule builder) in the search of an improving solution. In *steepest descent hill-climbing*, all neighbours are evaluated, the best one is selected and it replaces the original solution if it is an improving neighbour (in our case, with a smaller $E[C_{max}]$). In *simple hill-climbing*, evaluation stops as soon as the first improving neighbour is found. Local search starts again from that improving neighbour, so the procedure finishes when no neighbour satisfies the acceptation criterion.

Clearly, a central element in any local search procedure is the definition of neighbourhood. For the crisp job shop,

a well-known neighbourhood, which relies on the concepts of critical path and critical block, is that proposed in (Van Laarhoven, Aarts, & Lenstra 1992).

Definition 2. *A* move *for a feasible task processing order $\sigma \in \Sigma$ is defined as the change in the order of a pair of consecutive tasks (x, y) in a critical block. The* neighbourhood *of σ is defined as the set of processing orders obtained from σ after applying all possible moves.*

This neighbourhood presents certain interesting characteristics in the crisp case. First, all neighbours considered represent feasible solutions and the reversal of a non-critical arc in the graph can never reduce the makespan. Therefore, it reduces the search space without any loss in the potential quality of solutions. Also, there exists a finite sequence of transitions leading from any given element to some globally optimal element. This is usually referred to as connectivity property in the literature and, as we shall argue in the sequel, is a desirable (but not general) property of neighbourhood structures.

New Criticality Model and Neighbourhood Structure

We now propose a new neighbourhood structure for the job shop with uncertain durations represented as TFNs. To do so, we give a new definition of criticality, based on the fact that all arithmetic operations used in the scheduling process are performed on the three defining points or components of the TFNs. This allows to decompose the solution graph with fuzzy durations in three parallel graphs with crisp durations.

Definition 3. *Let $\sigma \in \Sigma$ be a feasible task processing order and let $G(\sigma) = (V, A \cup R(\sigma))$ be the corresponding solution graph, where the cost of any arc $(x, y) \in A \cup R(\sigma)$ is a TFN representing the processing time p_x of task x. From $G(\sigma)$, we define the* parallel solution graphs *$G^i(\sigma)$, $i = 1, 2, 3$, which are identical to $G(\sigma)$ except for the cost of arc $(x, y) \in A \cup R(\sigma)$, which for graph $G^i(\sigma)$ will be the i-th defining point of p_x, that is, p^i_x.*

Each of the parallel graphs $G^i(\sigma)$ is a solution graph identical to those for crisp JSP. Therefore, in each of them a critical path is the longest path from node *start* to node *end*. Notice that it is not necessarily unique; for instance, Figure 3, shows the three parallel graphs generated from the graph in Figure 2; we see that if $p^3_6 = 3$ there would be two critical paths in $G^3(\sigma)$: (3 4 6) and (3 5 6).

Using the parallel graph representation, we may define criticality for the fuzzy job shop as follows:

Definition 4. *A path P in $G(\sigma)$ is a* critical path *if and only if P is critical in some $G^i(\sigma)$. Nodes and arcs in a critical path are termed* critical. *A critical path is naturally decomposed into critical blocks $B_1, \ldots, B_r$, where a* critical block *is a maximal subsequence of tasks of a critical path requiring the same machine. Hence, the sets of critical paths, arcs, tasks and blocks are respectively the union of critical paths, arcs, tasks and blocks in the parallel solution graphs.*

Unlike for the crisp JSP, we may not state that the makespan of the schedule is the cost of a critical path. However, it holds that each component of the makespan $C^i_{max}(\sigma)$ is the cost of a critical path in the corresponding solution parallel graph $G^i(\sigma)$.

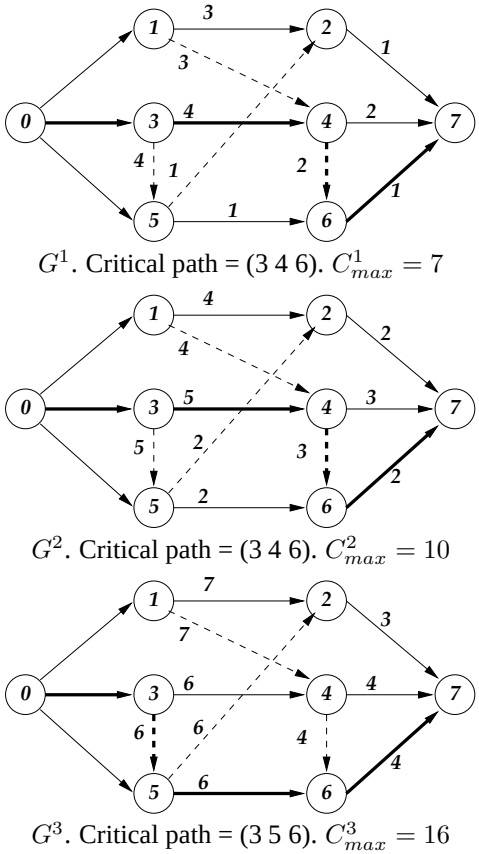

Figure 3: Parallel graphs corresponding to the graph in Figure 2, with critical paths in bold.

For a solution graph $G(\sigma)$ and a task x, let S_x and C_x denote respectively the starting and completion times of x, let $P\nu_x$ and $S\nu_x$ denote the predecessor and successor nodes of x on the machine sequence (predecessor and successor in $R(\sigma)$), and let PJ_x and SJ_x denote respectively the predecessor and successor nodes of x on the job sequence (predecessor and successor in A). We now extend some well-known notions for the crisp job shop and define the *head* r_x of task x as the starting time of x i.e. S_x, in our context, a TFN $r_x = (r^1_x, r^2_x, r^3_x)$ given by:

$$r_x = (r_{PJ_x} + p_{PJ_x}) \vee (r_{P\nu_x} + p_{P\nu_x}), \qquad (10)$$

and the *tail* q_x of task x as:

$$q_x = (q_{SJ_x} + p_{SJ_x}) \vee (q_{S\nu_x} + p_{S\nu_x}), \qquad (11)$$

Clearly, $C_x = r_x + p_x$. For each parallel graph $G^i(\sigma)$, r^i_x is the length of the longest path from node *start* to node x and $q^i_x + p^i_x$ is the length of the longest path from node x to node *end*. Hence, $r^i_x + p^i_x + q^i_x$ is the length of the longest path from node *start* to node *end* through node x in $G^i(\sigma)$; it is a lower bound of the i-th component of the makespan, being equal to $C^i_{max}(\sigma)$ if node x belongs to a critical path in $G^i(\sigma)$. For two TFNs A and B let $A \simeq B$ denote that

$\exists i, a^i = b^i$. The next proposition states some properties of critical arcs and tasks which follow trivially from the above definition:

Proposition 1. *If an arc (x, y) is critical, then $r_x + p_x \simeq r_y$. A task x is critical if and only if $r_x + p_x + q_x \simeq C_{max}$.*

For the proposed definition of criticality, we may extend the neighbourhood structure defined in (Van Laarhoven, Aarts, & Lenstra 1992) to the fuzzy case as follows:

Definition 5. *Given an arc $v = (x, y) \in R(\sigma)$, let $\sigma_{(v)}$ denote the processing order obtained from σ after an exchange in the processing order of tasks in arc v. Then, the* neighbourhood structure *obtained from σ is given by $H(\sigma) = \{\sigma_{(v)} : v \in R(\sigma)$ is critical$\}$.*

Comparison to Previous Approaches

In (Fortemps 1997), we find a first proposal to extend the neighbourhood structure in (Van Laarhoven, Aarts, & Lenstra 1992) to the fuzzy case. An arc (x, y) in the solution graph for the job shop problem with fuzzy durations is taken to be a *critical arc* if and only if $C_x \simeq S_y$, that is, if it is critical in any of its components. This definition, being quite natural, yields some counterintuitive results. For instance, in the graph from Figure 2, arc $(5, 2)$ would be considered to be critical.

In (González Rodríguez, Vela, & Puente 2007), this definition was refined by incorporating backpropagation: a *critical path* is taken to be a path in the solution graph from node 0 to node z where $C_z \simeq C_{max}$ and such that for all arcs (x, y) in the path it holds $C_x \simeq S_y$. A *critical arc* (respectively a *critical task*) is an arc (respectively task) belonging to a critical path.

Let us denote as critical' an arc which is critical according to this last definition. All arcs considered to be critical according to (Fortemps 1997) will still be critical', but the reverse does not hold, avoiding some of the counterintuitive examples. For instance, in Figure 2 the critical' arcs are (1,4), (3,4), (3,5), (4,6) y (5,6) and the critical' tasks are 1, 3, 4, 5 y 6. Clearly, it is possible to use the criticality definition from (González Rodríguez, Vela, & Puente 2007) to extend the neighbourhood structure proposed in (Van Laarhoven, Aarts, & Lenstra 1992) to the fuzzy case; for a given feasible task processing order σ, let $H'(\sigma) = \{\sigma_{(v)} : v \in R(\sigma)$ is critical'$\}$ denote the resulting neighbourhood. The next proposition follows trivially from the above definitions:

Proposition 2. *The set of critical paths is a subset of the critical' ones. Consequently, neighbourhood H is contained in H', that is*

$$\forall \sigma \in \Sigma, H(\sigma) \subseteq H'(\sigma) \tag{12}$$

Notice that the set of critical paths may be a proper subset of the set of critical' paths. Moreover, the second condition in Proposition 1 may not hold for a critical' task. For instance, in the example from Figure 2, arc $(1, 4)$ is critical' but is not critical and task 1 is critical' but $\forall i, r_1^i + p_1^i + q_1^i \neq C_{max}^i$.

Proposition 2 means a reduction in the number of neighbours when H is considered instead of H'. We now show that this reduction does not affect the potential quality of solutions found by the local search, as the reversal of a non critical arc can never reduce the makespan:

Proposition 3. *Let $\sigma \in \Sigma$ be a feasible processing order and let $\tau \in \Sigma$ be a feasible processing order obtained by the reversal an arc which is not critical in $G(\sigma)$. Then*

$$\forall i, \quad C_{max}^i(\sigma) \leq C_{max}^i(\tau) \tag{13}$$

and therefore

$$C_{max}(\sigma) \leq_E C_{max}(\tau) \tag{14}$$

Proof Let τ be obtained from σ by the reversal of an arc $v = (x, y)$ where v is not a critical arc, that is, it is not a critical arc in any of the parallel graphs $G^i(\sigma)$. Clearly, for all $i = 1, 2, 3$ the arcs inside critical paths of $G^i(\sigma)$ remain unchanged in $G^i(\tau)$ after the move, and therefore $C_{max}^i(\sigma) \leq C_{max}^i(\tau)$ for all i. □

In particular, neighbours in H' which do not belong to H can never improve the makespan:

Corollary 1. $\forall \sigma \in \Sigma, \forall \tau \in (H'(\sigma) - H(\sigma)) \bigcap \Sigma$, $C_{max}(\sigma) \leq_E C_{max}(\tau)$

We may conclude that using H instead of H' in the local search procedure leads to a reduction in the number of evaluated neighbours without missing any improving neighbours, i.e. with no loss in the quality of the solution obtained by the local search. Notice that if H' were to denote the neighbourhood proposed in (Fortemps 1997), all the above results would still hold.

Some Desirable Properties

Now we further study the new neighbourhood H, in order to prove that it has two highly desirable properties: feasibility and connectivity.

Theorem 1. *Let $\sigma \in \Sigma$ be a feasible task processing order; the reversal of a critical arc $v = (x, y) \in R(\sigma)$ produces a feasible processing order, i.e., $\sigma_{(v)} \in \Sigma$. In consequence, $H(\sigma) \subset \Sigma$.*

Sketch of Proof Suppose by contradiction that $G(\sigma_{(v)})$ has a cycle; this means that there exists an alternative path from x to y in $G(\sigma)$. This, together with (x, y) being critical, leads to the inequality $r_x^k + p_x^k \geq r_x^k + p_x^k + p_{SJ_x}^k + p_{PJ_y}^k$ for some $k = 1, 2, 3$, which contradicts the fact that all task durations are strictly positive. □

Notice that feasibility means that local search is automatically limited to the subspace of feasible task orders. It has the additional advantage of avoiding feasibility checks for the neighbours, hence increasing the efficiency of the local search procedure (reducing computational load) and avoiding the loss of feasible solutions that is usually encountered for feasibility checking procedures (cf. (Dell' Amico & Trubian 1993)).

Theorem 2. *H verifies the connectivity property, that is, for every non-optimal task processing order σ we may build a finite sequence of transitions of H leading from σ to a globally optimal processing order σ_0.*

Sketch of proof. The first step is to prove that if $V_\sigma(\sigma_0) = \{v = (x,y) \in R(\sigma) : v \text{ is critical }, (y,x) \in \overline{R(\sigma_0)}\} = \emptyset$ then σ is optimal, where $\overline{R(\sigma_0)}$ denotes the transitive closure of $R(\sigma_0)$. To do this, first prove that if σ is not optimal, there exists at least a critical arc in $R(\sigma)$. Then prove that if σ is not optimal, there exists some critical arc $(x,y) \in R(\sigma)$ such that $(y,x) \in \overline{R(\sigma_0)}$, using the fact that the set of critical arcs in $G(\sigma)$ is the union of critical arcs in $G^i(\sigma)$ for all i. The second step is to define the sequence $\{\lambda_k\}_{k\geq 0}$, where $\lambda_0 = \sigma$, and λ_{k+1} is obtained from λ_k by reversal of an arc $v \in V_{\lambda_k}(\sigma_0)$. To prove that this sequence is finite, define $\overline{M_\sigma(\sigma_0)} = \{v = (x,y) \in R(\sigma) : (y,x) \in \overline{R(\sigma_0)}\}$ and $\overline{M_\sigma(\sigma_0)} = \{v = (x,y) \in \overline{R(\sigma)} : (y,x) \in \overline{R(\sigma_0)}\}$, notice that if $\|\overline{M_{\lambda_k}(\sigma_0)}\| > 0$ then $\|\overline{M_{\lambda_{k+1}}(\sigma_0)}\| = \|\overline{M_{\lambda_k}(\sigma_0)}\| - 1$ and use the property from step 1. □

As mentioned above, connectivity is an important property for any neighbourhood used in local search. It ensures the non-existence of starting points from which local search cannot reach a global optimum. It also ensures asymptotic convergence in probability to a globally optimal order. Additionally, although the neighbourhood structure is used in a heuristic procedure in this paper, the connectivity property would allow to design an exact method for fuzzy job shop, like the well-known Branch and Bound for the crisp case.

Experimental Results

Hybrid methods combining a genetic algorithm with local search usually improve the quality of results obtained when these methods are used independently (see for instance (Ishibuchi & Murata 1998), (González Rodríguez, Vela, & Puente 2007), (Vela, Varela, & González 2008)). The usual approach is to apply local search to every chromosome right after this chromosome has been generated. The resulting algorithm is called a *memetic algorithm* (MA). To obtain experimental results for the proposed neighbourhood we shall then use a memetic algorithm proposed in (González Rodríguez, Vela, & Puente 2007) for the fuzzy job shop, but substituting the local search procedure used therein with the local search that results from using the new neighbourhood H together with simple hill-climbing.

The goal of the first set of experiments is to analyse the performance of the MA using the new local search compared to other methods from the literature. First, we consider a simulated annealing (SA) method from (Fortemps 1997), where the author generates fuzzy versions of well known benchmark problems (Applegate & Cook 1991): FT06 of size 6×6 and LA11, LA12, LA13 and LA14 of size 20×5. The only difference is that we generate 3-point fuzzy numbers or TFNs as fuzzy durations, instead of 6-point fuzzy numbers. The durations are symmetric TFNs, so the optimal solution to the crisp problem provides a lower bound for the fuzzified version (see (Fortemps 1997) and (González Rodríguez, Vela, & Puente 2007) for further detail). In total, a family of 10 instances of fuzzy job shop are generated from each original benchmark problem.

The MA has been run 30 times for each of the 50 fuzzy job shop instances with the following configuration: population size 20 and number of generations 50 (denoted 20/50 in the following). Table 1 shows the results obtained by the MA compared to the results SA reported in (Fortemps 1997). It contains expected makespan values of the best, average and worst solution for the 10 fuzzy instances of the same crisp problem. It can be seen that the MA performs better than the SA in all cases. In fact, the best expected makespan value obtained by the MA coincides with the optimal solution for the crisp problem. Regarding CPU times, on a 3GHz Pentium IV machine, our MA takes 0.18 seconds per run for FT06 and 1.45 in average for problems LA (no CPU times are reported for the SA in the above paper).

Table 1: MA versus SA

Problem	Method	Best	Avg	Worst
FT06	SA	55.02	55.2	56.01
(55)	MA	55	55.03	55.03
LA11	SA	1222	1222	1222
(1222)	MA	1222	1222	1222
LA12	SA	1041	1046.81	1056.35
(1039)	MA	1039	1039.67	1041.75
LA13	SA	1050	1155.07	1181.76
(1150)	MA	1150	1150	1150
LA14	SA	1292	1292	1292
(1292)	MA	1292	1292	1292

Now, we compare the MA with the GA proposed in (Sakawa & Kubota 2000), denoted GA00, according to the implementation and parameters used in (González Rodríguez *et al.* 2008). The problem instances are those proposed in (Sakawa & Kubota 2000), and comprise three problems of size 6×6, denoted S6-1, S6-2 and S6-3, and three problems of size 10×10, denoted S10-1, S10-2 and S10-3. Table 2 shows the expected makespan results obtained using the same parameter setting for MA as above, 20/50, and 100/200 for GA00, both ensuring the convergence of each algorithm. It is clear that MA outperforms GA00. Moreover, the CPU time is considerably reduced, going from 33.65 seconds per run to 1.91 seconds per run in average for problems of size 10×10, keeping convergence curves identical, both for the average population fitness and for the fitness of the best individual.

Another interesting comparison is that of the MA using the new local search with the MA from (González Rodríguez, Vela, & Puente 2007), denoted MA07, that uses H' as neighborhood structure and steepest descent as acceptance criterion. This comparison also serves to illustrate the theoretical results showing that H improves H' regarding the size of sets of neighbours without worsening the expected makespan values. The set of fuzzy problem instances are those used in (González Rodríguez, Vela, & Puente 2007) and they are generated from well-known benchmark problems for crisp JSP: FT10 of size 10×10, LA24 of size 10×15, FT20 of size 20×5 and ABZ7 of size 20×15, following the same method as above, so the optimal solution to the crisp problem provides a lower bound for the fuzzified version. The objective is to compare both MA and

Table 2: MA versus GA00

Problem	Method	Best	Avg	Worst
S6-1	GA00	82.25	83.11	86.25
	MA	79.75	79.75	79.75
S6-2	GA00	75.25	75.69	78.75
	MA	70.25	70.58	75.25
S6-3	GA00	66.25	66.25	66.25
	MA	66.25	66.25	66.25
S10-1	GA00	133.50	135.51	138.50
	MA	129.00	129.74	133.00
S10-2	GA00	127.50	128.38	132.75
	MA	123.75	126.67	127.75
S10-3	GA00	116.50	121.08	125.00
	MA	115.00	115.00	115.00

MA07 on larger (and more difficult) problems. In total, a family of 10 instances of fuzzy job shop are generated from each original benchmark problem.

Table 3 shows the average across the 10 instances of the same family of best, mean and worst solution obtained in 30 runs of each method with the parameter settings from (González Rodríguez, Vela, & Puente 2007). The optimal makespan of the crisp problem is shown between brackets under the problem's name. We see that the expected makespan values are almost identical for both methods, with differences between mean relative errors w.r.t. the expected makespan lower bound about 0.1% in average.

Table 3: Makespan comparison between MA and MA07.

Problem	Method	Best	Avg	Worst
FT10	MA07	934.05	938.29	952.78
(930)	MA	933.85	939.09	952.98
FT20	MA07	1165.35	1175.34	1180.90
(1165)	MA	1165.35	1175.12	1181.03
LA24	MA07	942.30	949.15	963.65
(935)	MA	942.93	948.56	962.65
ABZ7	MA07	677.90	686.18	693.38
(656)	MA	677.48	685.77	693.23

Table 4: Total Number of Neighbours and CPU Time for MA and MA07

Problem	Method	Neigh.	CPU
FT10	MA07	2.11E+06	50.45
	MA	9.43E+05	26.94
FT20	MA07	5.73E+06	125.87
	MA	2.26E+06	56.82
LA24	MA07	3.53E+06	122.26
	MA	1.29E+06	52.65
ABZ7	MA07	1.24E+07	852.45
	MA	3.25E+06	248.30

To further illustrate the efficiency gain of the new local search, combining neighbourhood H with simple hill climbing, Table 4 reports the total number of evaluated neighbours and the time required by each method, MA and MA07. The CPU times, on a 3GHz Pentium IV machine, refer to the seconds taken per run of the MA in average. There is a clear advantage in using the new local search. It is remarkable that the new neighbourhood H combined with the new selection criterion obtains identical makespan values to MA07, whilst reducing the number of evaluated neighbours more than 63% in average. This agrees with the theoretical properties regarding H and H'. The reduction of evaluated neighbours is reflected in a reduction of CPU times, despite it being more expensive to compute the neighbourhood H than H' because the search for critical arcs is performed through tree solution graphs instead of one. The average time reduction of MA w.r.t. MA07 is as much as 57%, with the remarkable case of the largest problem ABZ7, with a time reduction superior to 70%. Notice that the reduction in time is increased with the size of the problem, something natural since evaluating new neighbours has a higher computational cost in larger problems.

We have also conducted experiments (not reported here) to compare the gain using H instead H' independently from the acceptance criterion. The obtained makespan values equal those in Table 3, with an average reduction in the number of neighbours near 20%, which again increases with the problem size. These results are the same regardless of the acceptance criterion used, steepest descent and simple hill-climbing. Also, if MA is given the same time as MA07, it obtains slightly better solutions, but the improvement does not compensate the increase in running time.

The objective of the final set of experiments is to illustrate the synergy gained when combining the GA and the local search procedure (LS hereafter), by evaluating LS and GA separately, as well as both of them combined in the MA. This experiment also serves to test that there is no premature convergence to local optima. For results to be comparable, it is necessary to test all methods under equivalent conditions regarding computational resources. To do so, assuming a uniform computational load in all generations, for each

Table 5: Synergy: Comparison between LS, GA and MA.

Problem	Method	Best	Avg	Worst
	LS	965.50	989.13	1005.92
FT10	GA	937.40	956.19	981.45
(930)	MA	933.85	939.09	952.98
	LS	985.70	1004.70	1017.20
LA24	GA	960.80	981.53	1005.50
(935)	MA	942.93	948.56	962.65
	LS	1266.83	1298.85	1322.78
FT20	GA	1178.55	1188.36	1208.80
(1165)	MA	1165.35	1175.12	1181.03
	LS	724.00	734.59	742.13
ABZ7	GA	690.75	706.31	720.90
(655)	MA	677.48	685.77	693.23

problem we consider the ratio between CPU times of both GA and MA when they were executed with the same configuration (100/200) and then readjust the number of generations for the GA to obtain similar CPU times. To study LS independently, the LS algorithm is run 30 times for each instance starting from a set of 20000 random solutions, so as to evaluate the same number of individuals that the GA and obtain similar running times (about 27 seconds for FT10, 57 for FT20, 53 for LA24 and 248 for ABZ7). Table 5 shows the expected makespan results for each family of problems. It is worth noting that for MA, the CPU time dedicated to local search is 91-96% of the total, which can be explained by the fact that LS is applied to every chromosome in the population. We may conclude that, with the same computational time, the MA benefits from the synergy between the GA and LS and always yields better solutions than the GA or LS alone.

Conclusions

We have considered a job shop problem with uncertain processing times modelled as triangular fuzzy numbers, where the objective is to minimise the expected makespan value. We have proposed a new definition of criticality for this problem and a consequent new neighbourhood structure for local search. We have shown that the new structure improves previous proposals from the literature, reducing the set of neighbours by discarding non-improving ones. Additionally, we have shown that the new neighbourhood structure has two highly desirable properties: feasibility and connectivity, which further improve the efficiency of the resulting local search, as well as ensuring asymptotic convergence in probability to a globally optimal solution. The experimental results support the theoretical results and illustrate the potential of the new local search when used in combination with a genetic algorithm, comparing favourably with previous approaches from the literature.

In the future, makespan estimation procedures will be developed to reduce the computational cost of local search. These estimates may also be used to design a branch and bound type algorithm, taking advantage of the connectivity property. Additionally, metaheuristics other than the memetic algorithm will be proposed using the new neighbourhood, for instance, a taboo search procedure. Finally, the parallel solution graph model and consequent definitions of criticality and neighbourhood, proposed for durations represented as TFNs, may be extended to piece-wise linear fuzzy intervals, using α-cuts.

Acknowledgments

All authors are supported by MEC-FEDER Grant TIN2007-67466-C02-01.

References

Applegate, D., and Cook, W. 1991. A computational study of the job-shop scheduling problem. *ORSA Journal of Computing* 3:149–156.

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

Brucker, P., and Knust, S. 2006. *Complex Scheduling*. Springer.

Celano, G.; Costa, A.; and Fichera, S. 2003. An evolutionary algorithm for pure fuzzy flowshop scheduling problems. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 11:655–669.

Dell' Amico, M., and Trubian, M. 1993. Applying tabu search to the job-shop scheduling problem. *Annals of Operational Research* 41:231–252.

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

Dubois, D.; Fargier, H.; and Fortemps, P. 2003. Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge. *European Journal of Operational Research* 147:231–252.

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

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

González Rodríguez, I.; Vela, C. R.; and Puente, J. 2007. A memetic approach to fuzzy job shop based on expectation model. In *Proceedings of IEEE International Conference on Fuzzy Systems*, 692–697. London: IEEE.

Ishibuchi, H., and Murata, T. 1998. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics–Part C* 67(3):392–403.

Liu, B., and Liu, Y. K. 2002. Expected value of fuzzy variable and fuzzy expected value models. *IEEE Transactions on Fuzzy Systems* 10:445–450.

Sakawa, M., and Kubota, R. 2000. Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms. *European Journal of Operational Research* 120:393–407.

Słowiński, R., and Hapke, M., eds. 2000. *Scheduling Under Fuzziness*, volume 37 of *Studies in Fuzziness and Soft Computing*. Physica-Verlag.

Tavakkoli-Moghaddam, R.; Safei, N.; and Kah, M. 2008. Accessing feasible space in a generalized job shop scheduling problem with the fuzzy processing times: a fuzzy-neural approach. *Journal of the Operational Research Society* 59:431–442.

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

Vela, C. R.; Varela, R.; and González, M. A. 2008. Local search and genetic algorithm for the job shop scheduling problem with sequence dependent setup times. *Journal of Heuristics* Forthcoming.