ECAI 2010
H. Coelho et al. (Eds.)
IOS Press, 2010
© 2010 The authors and IOS Press. All rights reserved.
doi:10.3233/978-1-60750-606-5-739

Fast Local Search for Fuzzy Job Shop Scheduling

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

Abstract. In the sequel, we propose a new neighbourhood structure for local search for the fuzzy job shop scheduling problem. This is a variant of the well-known job shop problem, with uncertainty in task durations modelled using fuzzy numbers and where the goal is to minimise the expected makespan of the resulting schedule. The new neighbourhood structure is based in changing the relative order of subsequences of tasks within critical blocks. We study its theoretical properties and provide a makespan estimate which allows to select only feasible neighbours while covering a greater portion of the search space than a previous neighbourhood from the literature. Despite its larger search domain, experimental results show that this new structure notably reduces the computational load of local search with respect to the previous neighbourhood while maintaining or even improving solution quality.

1 INTRODUCTION

Scheduling problems form an important body of research since the late fifties, with multiple applications in industry, finance and science [22]. To reduce the gap between theory and practice, thus enhancing the range of applications, part of the research is devoted to model the uncertainty and vagueness pervading real-world situations [12]. 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 [5],[24]. They are also emerging as an interesting tool for improving solution robustness, a much-desired property in real-life applications [29],[14].

In deterministic 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 [1]. Some attempts have been made to extend these heuristic methods to the case 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 [2] and a genetic algorithm is hybridised with a local search procedure in [13]. For the job shop with different optimisation criteria, we find a neural approach [26], genetic algorithms [23],[20],[9], simulated annealing [7], genetic algorithms hybridised with local search [10] or particle swarm optimisation [18],[15].

In this paper, we intend to advance in the study of local search methods to solve the job shop problem with fuzzy durations where the goal is to minimise the expected makespan, denoted $FuzJ||E[C_{max}]$. We shall propose a new neighbourhood structure and see how it allows for finding same-quality solutions considerably faster than previous proposals.

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

2 THE FUZZY JOB SHOP SCHEDULING PROBLEM

The classical *job shop scheduling problem*, *JSP* in short, 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. There are also *capacity constraints*, whereby each task θ_{ij} requires the uninterrupted and exclusive use of one of the machines for its whole processing time. A feasible schedule is an allocation of starting times for each task such that all constraints hold. The objective is to find a schedule which is *optimal* according to some criterion, most commonly that the *makespan* is minimal.

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. However, based on previous experience, an expert may have some knowledge (albeit uncertain) about the duration. 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 an interval $[a^1, a^3]$ of possible values and a modal value a^2 in it. A TFN A, denoted $A = (a^1, a^2, a^3)$, has a membership function given by:

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

Triangular fuzzy numbers and more generally fuzzy intervals have been extensively studied in the literature (cf. [6]). A *fuzzy interval* Q is a fuzzy quantity (a fuzzy set on the reals) whose α-cuts $Q_\alpha = \{r \in \mathbb{R} : \mu_Q(r) \ge \alpha\}$, $\alpha \in (0, 1]$, are intervals (bounded or not). The *support* of Q is $Q_0 = \{r \in \mathbb{R} : \mu_Q(r) > 0\}$. A *fuzzy number* M is a fuzzy quantity whose α-cuts are closed intervals, denoted $M_\alpha = [\underline{m}_\alpha, \overline{m}_\alpha]$, 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*, in general cumbersome if not intractable. If f is a bivariate continuous isotonic function and M and N are two fuzzy numbers, $F = f(M, N)$ is another fuzzy number such that $F_\alpha = [f(\underline{m}_\alpha, \underline{n}_\alpha), f(\overline{m}_\alpha, \overline{n}_\alpha)]$. Computing the function is then equivalent to computing it on every α-cut. Both the addition and the maximum are continuous isotonic functions, so this equality may be applied to compute them. However, this still requires evaluating two sums or two maxima for every value $\alpha \in [0, 1]$. For the sake of simplicity and tractability of numerical calculations, the results of these operations will be approximated by a linear interpolation, evaluating only the operation on the three defining points of each TFN; this is a usual approach in the literature, taken, for instance, in [3],[7],[10],[18] or [21]. The approximated sum coincides with the actual sum, so for any pair of TFNs M and N:

$$M + N = (m^1 + n^1, m^2 + n^2, m^3 + n^3) \qquad (2)$$

Regarding the maximum, for any two TFNs M, N, if $F = \max(M, N)$ denotes their maximum and $G = (\max\{m^1, n^1\}, \max\{m^2, n^2\}, \max\{m^3, n^3\})$ 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, but 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_Q of a fuzzy quantity Q can be viewed as a possibility distribution on the real numbers; this allows to define the *expected value* of a fuzzy quantity [16], given for a TFN A by

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

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 [5]. It induces a total ordering $\leq_E$ in the set of fuzzy intervals [7], 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 $A \leq_E B$.

2.2 Disjunctive graph model

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 0 to the first task of each job and arcs form the last task of each job to node $nm+1$). Arcs in D are called *disjunctive arcs* and represent capacity constraints; $D = \cup_{i=1,\dots,m} D_i$, where D_i corresponds to machine M_i and includes two arcs (x, y) and (y, x) for each pair x, y 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 processing order of tasks π corresponds to an acyclic subgraph $G(\pi) = (V, A \cup R(\pi))$ of G, where $R(\pi) = \cup_{i=1\dots m} R_i(\pi)$, $R_i(\pi)$ being a hamiltonian selection of D_i. Using forward propagation in $G(\pi)$, we can obtain the starting and completion times for all tasks and, therefore, the makespan $C_{max}(\pi)$.

Since task processing times are fuzzy intervals, the addition and maximum operations used to propagate constraints are taken to be the corresponding operations on fuzzy intervals, approximated for the particular case of TFNs as explained above. The obtained schedule will be a fuzzy schedule in the sense that the starting and completion times of all tasks and the makespan are fuzzy intervals, interpreted 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.

2.3 Expected makespan

We have stated the goal of the job shop problem as finding a schedule which is optimal in the sense that the makespan is minimal. However, neither the maximum nor its approximation define a total ordering in the set of TFNs. In a similar approach to stochastic scheduling, it is possible to use the concept of expected value for a fuzzy quantity and the total ordering it provides, so the objective is to minimise the expected makespan $E[C_{max}(\sigma)]$, a crisp objective function. The resulting problem may be denoted $FuzJ||E[C_{max}]$, following the $\alpha|\beta|\gamma$ notation.

2.4 Criticality

In the crisp case, a *critical path* is defined as the longest path in a solution graph 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. [5]). 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).

In [11], a definition of criticality is proposed 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. Let $G(\pi) = (V, A \cup R(\pi))$ be a solution graph, where the cost of any arc $(x, y) \in A \cup R(\pi)$ is a TFN representing the processing time p_x of task x. From $G(\pi)$, we obtain the *parallel solution graphs* $G^i(\pi)$, $i = 1, 2, 3$, with identical structure to $G(\pi)$ but where the cost of any arc (x, y) is p^i_x, the i-th component of p_x. Since durations in each parallel graph $G^i(\pi)$ are deterministic, a critical path in $G^i(\pi)$ is undoubtedly the longest path from node *start* to node *end*. Notice that it is not necessarily unique.

Definition 1 *A path P in $G(\pi)$ is a* critical path *if and only if P is critical in some $G^i(\pi)$. Nodes and arcs in a critical path are termed* critical. *A critical path is naturally decomposed into critical blocks $B_1, \dots, B_r$, where a* critical block *is a maximal subsequence of tasks of a critical path requiring the same machine.*

Clearly, the sets of critical paths, arcs, tasks and blocks in $G(\pi)$ are respectively the union of critical paths, arcs, tasks and blocks in the parallel solution graphs. The makespan of the schedule is not necessarily the cost of a critical path, but each component $C^i_{max}(\pi)$ is the cost of a critical path in the solution parallel graph $G^i(\pi)$.

For a solution graph $G(\pi)$ and a task x, let $P\nu_x$ and $S\nu_x$ denote the predecessor and successor nodes of x on the machine sequence (in $R(\pi)$) and let PJ_x and SJ_x denote the predecessor and successor nodes of x on the job sequence (in A). The *head* of task x is $r_x = \max\{r_{PJ_x} + p_{PJ_x}, r_{P\nu_x} + p_{P\nu_x}\}$, the starting time of x, and the *tail* of task x is $q_x = \max\{q_{SJ_x} + p_{SJ_x}, q_{S\nu_x} + p_{S\nu_x}\}$, the time lag between x's completion and the end of all tasks (TFNs in our framework). The makespan coincides with the head of the last task and the tail of the first task: $C_{max} = r_{nm+1} = q_0$. Also, for each parallel graph $G^i(\pi)$, r^i_x is the length of the longest path from node 0 to node x, $q^i_x + p^i_x$ is the length of the longest path from node x to node $nm+1$, and $r^i_x + p^i_x + q^i_x$ is the length of the longest path from node 0 to node $nm+1$ through node x: it is a lower bound for $C^i_{max}(\pi)$, being equal if node x belongs to a critical path in $G^i(\pi)$.

3 FAST 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 solution, calculates its neighbourhood and then neighbours are evaluated in the search of an improving solution. In simple hill-climbing, the first improving neighbour found will replace the original solution, so local search starts again from that improving neighbour. The procedure finishes when no neighbour satisfies the acceptation criterion. Clearly, a central element in any local search procedure is the definition of neighbourhood.

3.1 Previous approaches

A well-known neighbourhood for the deterministic job shop is that proposed in [27]. Given a task processing order π, its neighbourhood structure is obtained by reversing all the critical arcs in $G(\pi)$. This structure was first extended to the fuzzy case in [7], where an arc (x, y) was taken to be critical in $G(\pi)$ if exists $i = 1, 2, 3$ such that $r_x^i + p_x^i = q_y^i$, i.e, the completion time of x coincides with the starting time of y in one component; the resulting neighbourhood will be denoted $\mathcal{N}_0$ in the following.

A second extension to the fuzzy case was proposed in [11], using the definition of criticality based on parallel solution graphs instead. Let us denote the resulting neighbourhood by $\mathcal{N}_1$. As a consequence of the criticality definitions, $\mathcal{N}_1 \subset \mathcal{N}_0$ and any neighbour $\sigma \in \mathcal{N}_0 - \mathcal{N}_1$ can never improve the expected makespan of the original solution. Additionally, all neighbours in $\mathcal{N}_1$ are feasible and the connectivity property holds: starting from any solution, it is possible to reach a given global optimum in a finite number of steps using this structure. The experimental results endorsed the good theoretical behaviour, obtaining better expected makespan values than previous approaches from the literature. However, the large size of the structure for the fuzzy case resulted in an extremely high computational load.

To improve on efficiency, a reduced structure, denoted $\mathcal{N}_2$ in the following, was proposed in [10], inspired in the proposal for the deterministic problem from [19]. The neighbourhood was based on reversing only those critical arcs at the extreme of critical blocks of a single path, so $\mathcal{N}_2 \subset \mathcal{N}_1$. Clearly, $\mathcal{N}_2$ contains only feasible neighbours, although connectivity fails to hold. It was proved that the reversal of a critical arc (x, y) can only lead to an improvement if (x, y) is at the extreme of a critical block, and therefore, all neighbours from $\mathcal{N}_1 - \mathcal{N}_2$ are non-improving solutions. The experimental results showed how $\mathcal{N}_2$ resulted in a much more efficient search obtaining the same expected makespan values as with $\mathcal{N}_1$. However, due to the fact that arcs may be critical on three different components, the neighbourhood size is still quite large and there is still room for improvement. It is also interesting to define different structures which allow for searching in different areas of the solution space.

3.2 New neighbourhood definition

All the neighbourhood structures proposed up to date are based on reversing a single critical arc. In the following, we propose a new neighbourhood structure obtained by "inverting more than one arc", that is, permuting the relative ordering of more than two consecutive tasks within a critical block, a proposal inspired in the work for deterministic job shop from [4].

Definition 2 *Let π be a task processing order and let $x, y) \in R(\pi)$ be a critical arc in the associated graph $G(\pi)$. The* neighbourhood structure *$\mathcal{N}_3(\pi)$ is obtained by considering all possible permutations of the sequences $(P\nu_x, x, y)$ and $(x, y, S\nu_y)$ where the relative order between x and y is reversed.*

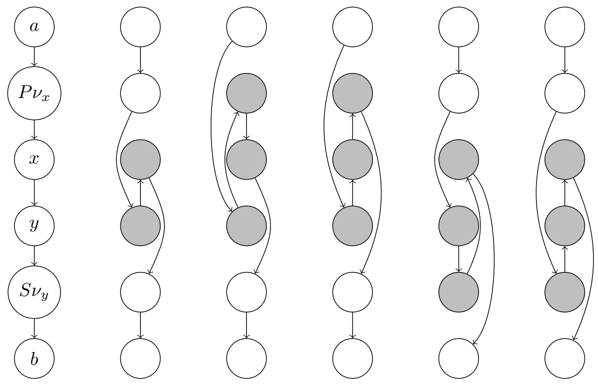

Figure 1. Representation of $\mathcal{N}_3^R$

For the aforementioned structures it is clear that $\mathcal{N}_2 \subset \mathcal{N}_1 \subset \mathcal{N}_3$. Since connectivity holds for $\mathcal{N}_1$, we automatically obtain the following result:

Theorem 1 *$\mathcal{N}_3$ verifies the connectivity property: given a globally optimal processing order π_0, it is possible to build a finite sequence of transitions of $\mathcal{N}_3$ starting from any non-optimal task processing order π and leading to π_0.*

Notice however that the considerations reported in [17] for the deterministic job shop are applicable here, making it advisable that $\mathcal{N}_3$ be reduced. Indeed, the reversal of a critical arc (x, y) can only lead to an improvement if at least one of $P\nu_x$ and $S\nu_y$ is non-critical, i.e., if (x, y) is at the extreme of a critical block [10]. Thus, depending on the critical block structure, at most three permutations should be taken into consideration for a neighbouring candidate. This motivates the definition of the following reduced neighbourhood:

Definition 3 *Let π be a task processing order and let $v = (x, y) \in R(\pi)$ be an arc at the extreme of a critical block in the associated graph $G(\pi)$. Then, the* reduced neighbourhood structure *$\mathcal{N}_3^R(\pi)$ is obtained as follows: if (x, y) is the only arc in the critical block, then (x, y) is reversed; if $P\nu_x$ is also critical (and $S\nu_y$ is not), then we consider all possible permutations of $(P\nu_x, x, y)$ where (x, y) is reversed; else, if $S\nu_y$ is critical, then we consider all possible permutations of $(x, y, S\nu_y)$ where (x, y) is reversed.*

It follows from the definition that $\mathcal{N}_2 \subset \mathcal{N}_3^R$. Table 1 shows the three cases that may be distinguished depending on the critical block structure, referred to as "small block", "begin block" and "end block". The notation used to refer to the resulting neighbours is introduced in the last column of the table.

The possible permutations are further illustrated in Figure 1: the first graph represents the machine sequence as it appears in π and the remaining graphs represent the five possible neighbours σ_1 to σ_5, where a and b represent tasks before and after the considered block in the machine sequence and the nodes in gray represent those tasks whose relative order is modified by $\mathcal{N}_3^R$. For the sake of simplicity, job arcs are omitted, as well as possible alternative paths generating cycles. It is nevertheless important to remark that such cycles may exist in neighbours σ_2 to σ_5 (this is not the case for σ_1, which is always feasible [11]). We shall see in the following how to select only feasible neighbours using an expected makespan estimate.

Table 1. Permutations considered in $\mathcal{N}_3^R$ and resulting neighbours

initial block		permutations	neighbour
small block	(x,y)	(y,x)	$\sigma_1=\pi_{(y,x)}$
begin block	$(P\nu_x,x,y)$	$(P\nu_x,y,x)$	$\sigma_1=\pi_{(y,x)}$
		$(y,P\nu_x,x)$	$\sigma_2=\pi_{(y,P\nu_x,x)}$
		$(y,x,P\nu_x)$	$\sigma_3=\pi_{(y,x,P\nu_x)}$
end block	$(x,y,S\nu_y)$	$(y,x,S\nu_y)$	$\sigma_1=\pi_{(y,x)}$
		$(y,S\nu_y,x)$	$\sigma_4=\pi_{(y,S\nu_y,x)}$
		$(y,x,S\nu_y)$	$\sigma_5=\pi_{(y,x,S\nu_y)}$

3.3 Makespan estimate and feasibility

In the local search procedure, only those neighbours with improving makespan are of interest. Thus, a makespan lower bound may help reduce the computational cost of local search by discarding uninteresting neighbours without actually evaluating them. For the case when only one arc (x,y) is reversed, $\sigma_1=\pi_{(y,x)}$, a lower bound of the neighbour's makespan may be obtained by computing the length of the longest path in $G(\sigma_1)$ containing either x or y [25]. This can be done quickly (in time $O(nm)$) using heads and tails. We now extend this idea to every neighbour σ in $\mathcal{N}_3^R(\pi)$, by computing the length of a longest path in $G(\sigma)$ containing at least one of the nodes involved in the move.

Let $X=\{x_1,\dots,x_s\}$ be the set of tasks whose relative order has been permuted to obtain σ from a feasible processing order π, i.e., $\sigma=\pi_{(x_1,\dots,x_s)}$ and let us assume that σ is feasible. Let r and q denote the heads and tails in $G(\pi)$ (before the move) and let r' and q' denote the heads and tails in $G(\sigma)$ (after the move). If σ is feasible, then $r'_x=r'_x$ for all predecessors of x_1 in σ and $q'_x=q'_x$ for all successors of x_s in σ. This suggests the following method lpath for computing the length of the longest path containing at least one task from X:

```
METHOD $\mathrm{lpath}(s,X)$
$a=x_1$;
$r'_a=\max\{r_{PJ_a}+p_{PJ_a}, r_{P\nu'_a}+p_{P\nu'_a}\}$;
for $i=2$ to $s$ do
    $b=x_i$;
    $r'_b=\max\{r_{PJ_b}+p_{PJ_b}, r'_a+p_a\}$;
    $a=b$;
$b=x_s$;
$q'_b=\max\{q_{SJ_b}+p_{SJ_b}, r_{S\nu'_b}+p_{S\nu'_b}\}$;
for $i=s-1$ to $1$ do
    $a=x_i$;
    $q'_a=\max\{q_{PJ_a}+p_{PJ_a}, q'_b+p_b\}$;
    $b=a$;
return $\max_{i=1,\dots,s}\{E[r'_{x_i}+p_{x_i}+q'_{x_i}]\}$;
```

Method $\mathrm{lpath}(s,X)$ provides an inexpensive lower bound for the expected makespan of a feasible neighbour $\sigma=\pi_{(X)}$. If σ is unfeasible, the method is still applicable even if in this case it makes no sense to talk about makespan nor lower bounds thereof. Additionally, this method may be used to select the most promising neighbour as follows:

```
METHOD $\mathrm{estim}(x,y)$
$e_1=\mathrm{lpath}(2,(y,x))$;
if $P\nu_x$ is critical then
    $e_2=\mathrm{lpath}(3,(y,P\nu_x,x))$;
    $e_3=\mathrm{lpath}(3,(y,x,P\nu_x))$;
else if $S\nu_y$ is critical then
    $e_4=\mathrm{lpath}(3,(y,S\nu_y,x))$;
    $e_4=\mathrm{lpath}(3,(S\nu_y,y,x))$;
return $\sigma_i$ such that $i=\min\{i:e_i\le e_j: i,j=1,\dots,5\}$;
```

Method estim not only allows selecting the most promising neighbour in an inexpensive manner, but also the produced neighbour will always be feasible.

Theorem 2 *Let π be a feasible task processing order and let (x,y) be an arbitrary critical arc. Then, the method* $\mathrm{estim}(x,y)$ *always returns a feasible processing order.*

Proof 1 *If π and (x,y) are such that the small block case holds, then the only resulting neighbour $\sigma_1=\pi_{(y,x)}$ is known to be feasible, as desired.*

Let us now suppose that the begin block case holds, i.e., $(P\nu_x,x,y)$ is at the beginning of a critical block; to simplify notation, let $P\nu_x$ be denoted by z in the following. There are three possible neighbours: $\sigma_1=\pi_(y,x)$, $\sigma_2=\pi_(y,z,x)$ and $\sigma_3=\pi_(y,x,z)$.

Without loss of generality, let us suppose that σ_2 is unfeasible, i.e., the sequence (y,z,x) generates a cycle in $G(\sigma_2)$, and let us prove that, in that case, $\mathrm{lpath}(2,(y,x))\le\mathrm{lpath}(3,(y,z,x))$. *The changes produced to transform $G(\pi)$ into $G(\sigma_2)$ are the following: the processing order (z,x,y) changes to (y,z,x) and, therefore arcs $(a,z),(x,y),(y,s)$ disappear, having $(a,y),(y,p),(x,s)$ instead. Since the reversal of (x,y) cannot produce a cycle, unfeasibility can only be produced by the existence of an alternative path both in $G(\pi)$ and $G(\sigma_2)$ going from z to y through SJ_z and PJ_y.*

Let r' and q' denote the heads and tails computed by $\mathrm{lpath}(2,(y,x))$ *in $G(\sigma_1)$ and let r'' and q'' denote the heads and tails computed by* $\mathrm{lpath}(3,(y,z,x))$ *in $G(\sigma_1)$. Clearly, $q'_x=q''_x$. Let us now see that it also holds $r'_y=r''_y$.*

Indeed, since there exists an alternative path in $G(\pi)$ from SJ_z to PJ_y, then

$$r_{PJ_y}\ge r_{SJ_z}+p_{SJ_z}\ge r_z+p_z+p_{SJ_z}>r_z+p_z$$

and therefore

$$r'_y=\max\{r_{PJ_y}+p_{PJ_y}, r_z+p_z\}=r_{PJ_y}+p_{PJ_y}$$

On the other hand, in $G(\pi)$ it holds that:

Therefore, if σ_2 is unfeasible, $\mathrm{lpath}(2,(y,x))\le\mathrm{lpath}(3,(y,z,x))$, *and in consequence* estim *will never return σ_2.*

The end block case is analogous.

The new neighbourhood using estim does therefore preserver feasibility; notice however that this is not the case for connectivity.

4 EXPERIMENTAL RESULTS

The purpose of this section is to provide an experimental evaluation of the proposed neighbourhood structure. To this end, we shall use a set of 120 problem instances proposed in [10], which result from fuzzifying 12 benchmark problems for job shop: the well-known FT10 (size 10×10) and FT20 (20×5), and La21, La24, La25 (15×10), La27, La29 (20×10), La38, La40 (15×15), and ABZ7, ABZ8, ABZ9 (20×15), a set of 10 problems considered to be hard to solve for classical job shop. There are ten fuzzy versions of each benchmark, generated following [7], so task durations become symmetric TFNs where the modal value is the original duration, thus ensuring that the optimal solution to the crisp problem provides a lower bound for the fuzzified version.

Plain hill-climbing algorithms cannot be expected to perform very well on complex problems. However, hybrid methods combining a genetic algorithm (GA) with local search (LS) generally improve the quality of results obtained when these methods are used independently (cf. [13], [8], [28]). The usual approach is to apply local search to every chromosome right after this chromosome has been generated, resulting in a so-called *memetic algorithm* (MA). We shall then test $\mathcal{N}_3$ by incorporating it to a MA from the literature, allowing for comparisons with previous neighbourhood structures. In [11], a MA was presented which used $\mathcal{N}_1$; this algorithm, denoted MA-1 in the following, compared favourably with previous approaches from the literature in terms of expected makespan optimisation (among others, it improved a simulated annealing algorithm using $\mathcal{N}_0$ from [7] and a memetic algorithm using a refinement of $\mathcal{N}_0$ from [8]). The results reported in [10] correspond to a variant of this memetic algorithm, using $\mathcal{N}_2$ as neighbourhood structure and incorporating a makespan lower bound to discard non-improving neighbours. The resulting algorithm, denoted MA-2 in the following, obtained expected makespan values identical to those from MA-1 (and therefore, better than previous approaches), but greatly improving on efficiency, with an average CPU time reduction of 76.49% w.r.t. MA-1. We shall therefore use MA-2 as baseline algorithm, substituting $\mathcal{N}_2$ and the used makespan estimate by $\mathcal{N}_3^R$ with the makespan estimate and pre-selection method given in estime.

Table 2. Results of MA-2 (200 generations) and MA-3 (140 generations)

Problem	MA	$E[C_{max}]$ Best	Avg	Worst	No. Neigh.	CPU
ft10	MA-2	934	938	953	1.20E+05	5.55
	MA-3	934	938	952	9.47E+04	4.55
ft20	MA-2	1165	1175	1181	1.72E+05	7.03
	MA-3	1165	1174	1179	1.37E+05	5.67
la21	MA-2	1056	1059	1063	1.92E+05	9.74
	MA-3	1056	1059	1061	1.48E+05	7.85
la24	MA-2	942	948	958	1.86E+05	9.47
	MA-3	942	947	958	1.42E+05	7.59
la25	MA-2	980	986	991	1.92E+05	9.65
	MA-3	980	985	990	1.47E+05	7.69
la27	MA-2	1254	1265	1270	3.26E+05	17.05
	MA-3	1252	1264	1269	2.55E+05	13.58
la29	MA-2	1182	1200	1220	2.97E+05	16.19
	MA-3	1180	1196	1211	2.33E+05	12.81
la38	MA-2	1214	1226	1248	2.67E+05	17.13
	MA-3	1213	1226	1249	2.06E+05	13.61
la40	MA-2	1234	1241	1253	2.81E+05	17.72
	MA-3	1233	1239	1246	2.19E+05	14.26
abz7	MA-2	677	685	693	4.33E+05	29.95
	MA-3	677	685	692	3.47E+05	24.10
abz8	MA-2	690	700	708	4.93E+05	32.51
	MA-3	687	698	706	3.93E+05	26.35
abz9	MA-2	703	715	726	4.43E+05	30.86
	MA-3	701	713	724	3.53E+05	24.87

A first set of results illustrates how MA-3, using $\mathcal{N}_3$, obtains equal or even better results in a faster way than MA-2, using $\mathcal{N}_2$. To this end, both algorithms are run with a different number of generations: 200 for MA-2 and 140 (30% less) for MA-3. Table 2 shows a summary of the obtained results. For each family of fuzzy instances obtained from the same deterministic problem and for each neighbourhood structure, it shows average values across 30 runs of the corresponding MA: best, average and worse expected makespan value, average number of evaluated neighbours and CPU time in seconds per run. The expected makespan values obtained using $\mathcal{N}_3^R$ is always slightly better than the values obtained with $\mathcal{N}_3^R$ in all cases, using approximately 20% less neighbours and CPU time.

To compare the behaviour of both neighbourhood structures when MA-3 is run with the same number of generations as MA-2, we have selected one of the largest problems: abz9-01, the first fuzzy instance of abz9. Table 3 shows how the better behaviour of $\mathcal{N}_3^R$ is maintained with a larger number of generations (200 and 500). Since $\mathcal{N}_3^R$ is larger $\mathcal{N}_2$, MA-3 should be expected to require more CPU time than MA-2 for the same number of generations. However, this increase is not linear in the number of generations. While for 200 generations MA-3 takes 13.5% more CPU time than MA-2, for 500 generations this increase is less 10%. This, together with the fact that MA-3 was reaching better solutions earlier, suggests that $\mathcal{N}_3^R$ drives the search to better areas in the solution space where the effort needed by the LS with $\mathcal{N}_3^R$ decreases at a greater pace than the effort needed when using $\mathcal{N}_2$.

Table 3. Results of MA-2 and MA-3 for 200 and 500 generations on instance abz9-01.

NoGen	LS	$E[C_{max}]$ Best	Avg	Worst	CPU
200	MA-2	704.75	715.43	729.75	30.03
	MA-3	697.00	708.46	718.75	34.07
500	MA-2	698.25	708.42	719.75	70.50
	MA-3	690.00	701.58	710.25	77.50

5 CONCLUSIONS

We have tackled a variant of the job shop scheduling problem where uncertainty in durations is modelled using triangular fuzzy numbers and where the objective is to minimise the expected makespan. We have proposed a new neighbourhood structure for local search, based on permuting the relative order of critical tasks, denoted $\mathcal{N}_3^R$, covering a greater portion of promising areas in the search space than previous proposals from the literature. We have also provided a method, lpath, to obtain a lower bound of the expected makespan of feasible neighbours, which is later used in a procedure estim in order to always select feasible neighbours. To obtain experimental results, the structure and selection method are incorporated to a simple hill-climbing local search and combined with a genetic algorithm. Thanks to its ability to explore alternative areas of the solution space, the resulting algorithm reaches the best solutions obtained by previous approaches much faster: in 30% less generations and 20% less CPU time than the best approach so far. In the future, this fast local search may be used in alternative meta-heuristics, such as taboo search, which have proved successful for other variants of the job shop and which were not fit for the less efficient neighbourhood structures previously proposed for fuzzy job shop.

REFERENCES

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

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

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

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

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

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

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

[8] I. González Rodríguez, C. R. Vela, and J. Puente, 'Sensitivity analysis for the job shop problem with uncertain durations and flexible due dates', *IWINAC 2007, Lecture Notes in Computer Science*, **4527**, 538–547, (2007).

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

[10] I. González Rodríguez, C. R. Vela, A. Hernández-Arauzo, and J. Puente, 'Improved local search for job shop scheduling with uncertain durations.', in *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS-2009)*, pp. 154–161, Thesaloniki, (2009). AAAI Press.

[11] I. González Rodríguez, C. R. Vela, J. Puente, and R. Varela, 'A new local search for the job shop problem with uncertain durations', in *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS-2008)*, pp. 124–131, Sidney, (2008). AAAI Press.

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

[13] H. Ishibuchi and T. Murata, 'A multi-objective genetic local search algorithm and its application to flowshop scheduling', *IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews*, **67**(3), 392–403, (1998).

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

[15] D. Lei, 'Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems', *International Journal of Advanced Manufacturing Technology*, **37**, 157–165, (2008).

[16] B. Liu and Y. K. Liu, 'Expected value of fuzzy variable and fuzzy expected value models', *IEEE Transactions on Fuzzy Systems*, **10**, 445–450, (2002).

[17] H. Matsuo, C.J. Suh, and R.S. Sullivan, 'A controlled search simulated annealing method for the general jobshop scheduling problem.', Working paper 03-44-88, Graduate School of Business, University of Texas, (1988).

[18] 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).

[19] E. Nowicki and C. Smutnicki, 'A fast taboo search algorithm for the job shop scheduling problem', *Management Science*, **42**(6), 797–813, (1996).

[20] S. Petrovic, C. Fayad, D. Petrovic, E. Burke, and G. Kendall, 'Fuzzy job shop scheduling with lot-sizing', *Annals of Operations Research*, **159**, 275–292, (2008).

[21] S. Petrovic and X. Song, 'A new approach to two-machine flow shop problem with uncertain processing times', *Optimization and Engineering*, **7**, 329–342, (2006).

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

[23] M. Sakawa and R. Kubota, '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, (2000).

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

[25] E. D. Taillard, 'Parallel taboo search techniques for the job shop scheduling problem', *ORSA Journal on Computing*, **6**(2), 108–117, (1994).

[26] R. Tavakkoli-Moghaddam, N. Safei, and M. M. O. Kah, '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, (2008).

[27] P. Van Laarhoven, E. Aarts, and K. Lenstra, 'Job shop scheduling by simulated annealing', *Operations Research*, **40**, 113–125, (1992).

[28] C. R. Vela, R. Varela, and M. A. González, 'Local search and genetic algorithm for the job shop scheduling problem with sequence dependent setup times', *Journal of Heuristics*, **16**(2), 139–165, (2010).

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