Combining Neighbourhoods in Fuzzy Job Shop Problems

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

[1] A.I. Centre and Department of Computer Science,
University of Oviedo, Spain
{puente,crvela}@uniovi.es
http://www.aic.uniovi.es/Tc

[2] Department of Mathematics, Statistics and Computing,
University of Cantabria, Spain
ines.gonzalez@unican.es

Abstract. In the sequel, we propose a new neighbourhood structure for local search for the fuzzy job shop scheduling problem, which is a variant of the well-known job shop problem, where uncertain durations are modelled as fuzzy numbers and the objective is to minimise the expected makespan of the resulting schedule. The new neighbourhood structure is based on changing the position of a task in a critical block. We provide feasibility conditions and a makespan estimate which allows to select only feasible and promising neighbours. The experimental results illustrate the success of our proposal in reducing expected makespan within a memetic algorithm. The experiments also show that combining the new structure with an existing neighbourhood from the literature considering both neighborhoods at the same time, provides the best results.

1 Introduction

Scheduling forms an important body of research since the late fifties, with multiple applications in industry, finance and science [16]. Traditionally, it has been treated as a deterministic problem that assumes precise knowledge of all data. However, modelling real-world problems often involves processing uncertainty, for instance in activity durations. In the literature we find different proposals for dealing with ill-known durations [11]. Perhaps the best-known approach is to treat them as stochastic variables. An alternative is to use fuzzy numbers or, more generally, fuzzy intervals in the setting of possibility theory, which is said to provide a natural framework, simpler and less data-demanding than probability theory, for handling incomplete knowledge about scheduling data (c.f. [4]).

The complexity of scheduling problems such as job shop means that practical approaches to solving them usually involve heuristic strategies [2]. Extending these strategies to problems with fuzzy durations in general requires a significant reformulation of both the problem and solving methods. Proposals from the literature include a neural approach [20], genetic algorithms [18],[15],[7], simulated annealing [5] and genetic algorithms hybridised with local search [6],[9].

In this paper, we intend to advance in the study of local search methods to solve the job shop problem with task durations given as triangular fuzzy numbers and where the goal is to minimise the expected makespan, denoted $FuzJ||E[C_{max}]$. In [17] a neighbourhood $\mathcal{N}_3$ is proposed that, embedded in a memetic algorithm, notably reduces the computational load of local search with respect to a previous neighbourhood while maintaining or even improving solution quality. We shall propose a new neighbourhood structure, based on a definition of criticality from [9]. This will allow to obtain better quality solutions at the cost of increasing the number of neighbours. Even better, when it is used in conjunction with $\mathcal{N}_3$ it reaches even better solutions with a smaller set of neighbours and hence with a lower computational load. Finally, we propose that the local search be also integrated into the genetic algorithm framework.

2 Job Shop Scheduling with Uncertain Durations

The *job shop scheduling problem*, also denoted *JSP*, consists in scheduling a set of jobs $\{J_1, \dots, J_n\}$ on a set of physical resources or machines $\{M_1, \dots, M_m\}$, subject to a set of constraints. There are *precedence constraints*, so each job J_i, $i = 1, \dots, n$, consists of m tasks $\{\theta_{i1}, \dots, \theta_{im}\}$ 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 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

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 is available. Such knowledge can be modelled using a *triangular fuzzy number* or TFN, given by an interval $[n^1, n^3]$ of possible values and a modal value n^2 in it. For a TFN N, denoted $N = (n^1, n^2, n^3)$, the membership function takes the following triangular shape:

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

In the job shop, we essentially need two operations on fuzzy numbers, 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 [5] and approximate the results of these operations, evaluating the operation only on the three defining points of each TFN. It turns out that for any pair of TFNs M

and N, the approximated sum $M + N \approx (m^1 + n^1, m^2 + n^2, m^3 + n^3)$ coincides with the actual sum of TFNs; this may not be the case for the maximum $\max(M, N) \approx (\max(m^1, n^1), \max(m^2, n^2), \max(m^3, n^3))$, although they have identical support and modal value.

The membership function of a fuzzy number can be interpreted as a possibility distribution on the real numbers. This allows to define its expected value [12], given for a TFN N by $E[N] = \frac{1}{4}(n^1 + 2n^2 + n^3)$. It coincides with the neutral scalar substitute of a fuzzy interval and the centre of gravity of its mean value [4]. It induces a total ordering $\leq_E$ in the set of fuzzy numbers [5], where for any two fuzzy numbers M, N $M \leq_E N$ if and only if $E[M] \leq E[N]$.

2.2 Fuzzy Job Shop Scheduling

A job shop problem instance may be represented by a directed graph $G = (V, A \cup D)$. V contains one node $x = m(i-1) + j$ per task θ_{ij}, $1 \leq i \leq n$, $1 \leq j \leq m$, plus two additional nodes 0 (or *start*) and $nm + 1$ (or *end*), representing dummy tasks with null processing times. Arcs in A, called *conjunctive arcs*, 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 node *end*). Arcs in D, called *disjunctive arcs*, represent capacity constraints; $D = \cup_{j=1}^{m} D_j$, 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 (x, y) is weighted with the processing time p_x of the task at the source node (a TFN in our case). A feasible task processing order σ is represented by a *solution graph*, an acyclic subgraph 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. Using forward propagation in $G(\sigma)$, it is possible to obtain the starting and completion times for all tasks and, therefore, the schedule and the makespan $C_{max}(\sigma)$.

The schedule will be fuzzy in the sense that the starting and completion times of all tasks and the makespan are TFNs, 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.

Given that the makespan is a TFN and neither the maximum nor its approximation define a total ordering in the set of TFNs, it is necessary to reformulate what is understood by "minimising the makespan". 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.

Another concept that needs some reformulation in the fuzzy case is that of criticality, an issue far from being trivial. In [5], an arc (x, y) in the solution graph is taken to be critical if and only if the completion time of x and the starting time of y coincide in any of their components. In [9], it is argued that this definition yields some counterintuitive examples and a more restrictive notion is proposed. From the solution graph $G(\sigma)$, three *parallel solution graphs* $G^i(\sigma)$, $i = 1, 2, 3$, are derived with identical structure to $G(\sigma)$, but where the cost of arc

$(x, y) \in A \cup R(\sigma)$ in $G^i(\sigma)$ is p_x^i, the i-th component of p_x. Each parallel solution graph $G^i(\sigma)$ is a disjunctive graph with crisp arc weights, so in each of them a critical path is the longest path from node *start* to node *end*. For the fuzzy solution graph $G(\sigma)$, a path will be considered to be *critical* if and only if it is critical in some $G^i(\sigma)$. Nodes and arcs in a critical path are termed critical and a critical path is naturally decomposed into critical blocks, these being maximal subsequences of tasks requiring the same machine.

In order to simplify expressions, we define the following notation for a feasible schedule. 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 the starting time of x, a TFN given by $r_x = \max\{r_{PJ_x} + p_{PJ_x}, r_{P\nu_x} + p_{P\nu_x}\}$, and the *tail* of task x is the time lag between the moment when x is finished until the completion time of all tasks, a TFN given by $q_x = \max\{q_{SJ_x} + p_{SJ_x}, q_{S\nu_x} + p_{S\nu_x}\}$.

3 Improved 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.

Neighbourhood structures have been used in different metaheuristics to solve the fuzzy job shop. In [5], a neighbourhood is used in a simulated annealing algorithm. The same neighbourhood is used in [6] for a memetic algorithm (MA) hybridising a local search procedure (LS) with a genetic algorithm (GA) using permutations with repetition as chromosomes. Results in [6] show that the hybrid method compares favourably with the simulated annealing from [5] and a GA from [18]. The same memetic algorithm is used in [9], but here the local search procedure uses the neighbourhood based on parallel graphs. The experimental results reported in [9] show that this new memetic algorithm performs better than state-of-the-art algorithms. Despite satisfactory, the results also suggest that the algorithm has reached its full potential and, importantly, most of the computational time it requires corresponds to the local search. In [8] and [17] two new neighbourhood stuctures have been defined. Both reduce the computational cost of the local search, specially the one from [17], while keeping similar or even identical quality of solutions.

In the following, we propose to improve local search efficiency in two steps. A first idea is to introduce in the local search algorithm a new neighbourhood structure, based on inserting a critical task into other position of its critical block, evaluating neighbours in a efficient manner and using makespan estimators.

A second idea, previously used in the crisp framework in [13], is to use the new structure together with a previous one to obtain an advanced neighbourhood definition which combines both their advantages.

3.1 Previous Approaches

A well-known neighbourhood for the deterministic job shop is that proposed in [21]. 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 [5], where a disjunctive 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 [9], 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 [8], inspired in the proposal for the deterministic problem from [14]. 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.

All these neighbourhood structures were based on reversing a single critical arc. In [17], 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, was proposed. Given a task processing order π and a critical arc (x, y) in the associated graph $G(\pi)$, $\mathcal{N}_3(\pi)$ 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.

For the aforementioned structures it is clear that $\mathcal{N}_2 \subset \mathcal{N}_1 \subset \mathcal{N}_3$. Moreover, $\mathcal{N}_3$ verifies the connectivity property and, as $\mathcal{N}_1$, contains many not-improving neighbours. A reduced neighbourhood, $\mathcal{N}_3^R$, is defined using only the extreme of critical blocks. $\mathcal{N}_3^R$ contains $\mathcal{N}_2$ while covering a greater portion of promising areas in the search space. In principle, $\mathcal{N}_3^R$ may contain unfeasible neighbours, so a method lpath is provided in [17] that allows to obtains a lower bound of the expected makespan of feasible neighbours, which is later used in order to always select feasible neighbours. Despite its larger search domain, this new structure notably reduces the computational load of local search with respect to the previous neighbourhood while maintaining solution quality.

3.2 New Neighbourhood Definition

All the neighbourhood structures proposed up to date are based on reversing one or more critical arcs. In the following, we propose a new neighbourhood structure obtained by inserting a critical task in another position in its critical block, a proposal inspired in the work for deterministic job shop from [3].

For a task x *inside* a block $b = (b', x, b'')$, where b' and b'' are sequences of tasks, the aim of the new neighbourhood is to move x to the first or the last position in b. Actually such moves may lead to infeasible solutions; if this is the case, we consider the closest move to the first or the last position for which feasibility is preserved.

Testing the feasibility of a solution may be computationally expensive. The next proposition, inspired in [3], gives a sufficient condition for feasibility after moving an operation x in a critical block towards the beginning of such block.

Proposition 1. *Let σ be a feasible processing order and let $b = (b'_1\ b'_2\ x\ b'')$ be a critical block in $G^i(\sigma)$ for some i, where b'_1, b'_2 and b'' are sequences of tasks, a sufficient condition for feasibility of a solution $\pi = \sigma_{(b'_1, x, b'_2, b'')}$ is that*

$$\exists j = 1, 2, 3, \quad r^j_{PJ_x} < r^j_{SJ_y} + p^j_{SJ_y} \quad \forall y \in b'_2 \tag{2}$$

The proof of this proposition follows from the fact of that feasibility is lost if a cycle in the resulting digraph exist, and this cycle can only exist if and only if there exists an alternative path from a task in b'_2 to PJ_x. This property suggests the following definition of neighbourhood.

Definition 1 **($\mathcal{N}_4(\pi)$).** *Let π be a task processing order and let x an operation in a critical block b. In a neighboring solution x is moved closest to the first or the last operation of b for which the sufficient condition of feasibility given by proposition 1 is preserved.*

Theorem 1. *$\mathcal{N}_4$ verifies the connectivity property: given a globally optimal processing order π_0, it is possible to build a finite sequence of transitions of $\mathcal{N}_4$ starting from any non-optimal task processing order π and leading to π_0.*

The proof of this property is ommited due to space constraints.

Notice however that the considerations reported in [10] about the so called *elimination properties* for the deterministic job shop are applicable here, making it advisable that $\mathcal{N}_4$ be reduced. Indeed, the insertion of a critical task x inside a block in other position can only lead to an improvement if the new position is at the extreme of the block. This motivates the definition of the following reduced neighbourhood:

Definition 2. *Let π be a task processing order and let x be a task in a critical block b in the associated graph $G(\pi)$. Then, in a neighboring solution of the* reduced neighbourhood structure*, $\mathcal{N}_4^R(\pi)$, x is moved to the first or the last operation of b whenever the sufficient condition of feasibility given by proposition 1 is preserved.*

3.3 Makespan Estimation

In a monotonic local search method, as hill climbing used in this work, only those neighbours with improving makespan are of interest. Hence a makespan estimation 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 [19]. This can be done quickly (in time $O(nm)$) using heads and tails. In [17] this idea has been extended 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. This is still valid for $\mathcal{N}_4^R$ if we consider the sequence of tasks $X = (x_1, \ldots, x_s)$ whose relative order has been permuted, although the method provides an estimate which is not necessarily a lower bound.

4 Experimental Results

We now consider 12 benchmark problems for job shop: the well-known FT10 and FT20, and the set of 10 problems identified in [1] as hard to solve for classical JSP: La21, La24, La25, La27, La29, La38, La40, ABZ7, ABZ8, and ABZ9. Ten fuzzy versions of each benchmark are generated following [5] and [9], so task durations become symmetric TFNs where the modal value is the original duration, ensuring that the optimal solution to the crisp problem provides a lower bound for the fuzzified version. In total, we consider 120 fuzzy job shop instances, 10 for each of the 12 crisp benchmark problems.

The goal of this section is to evaluate empirically our proposals. We consider the memetic algorithm (MA) presented in [17] which improved previous approaches from the literature in terms of makespan optimisation and efficiency. This algorithm combines a genetic algorithm with a simple hill-climbing local search procedure based on the neighbourhood structure $\mathcal{N}_3^R$. We shall use it as a baseline algorithm and introduce the different structures in the local search module: $\mathcal{N}_3^R$, $\mathcal{N}_4^R$ and also, following the work from [13] for deterministic JSP, $\mathcal{N}_3^R \cup \mathcal{N}_4^R$. We have run the MA using the same parameters as in [17] (population size 100 and 200 generations). Table 1 shows for each MA version the average

Table 1. Results of MA using $\mathcal{N}_3^R$, $\mathcal{N}_4^R$, and $\mathcal{N}_3^R \cup \mathcal{N}_4^R$. CPU times are seconds (C++, Xeon E5520 2.26GHz).

Problem	Size	MA($\mathcal{N}$)	$RE\_E[C_{max}]$			%Neigh.Inc	CPU
			Best	Avg	Worst		
FT10	10×10	$\mathcal{N}_3^R$	0.41	0.80	2.26		2.88
		$\mathcal{N}_4^R$	0.41	0.72	1.74	64.4	4.39
		$\mathcal{N}_3^R \cup \mathcal{N}_4^R$	0.41	0.70	1.78	48.5	4.19
FT20	20×5	$\mathcal{N}_3^R$	0.03	0.70	1.13		3.80
		$\mathcal{N}_4^R$	0.03	0.31	1.12	166.6	8.14
		$\mathcal{N}_3^R \cup \mathcal{N}_4^R$	0.03	0.43	1.12	104.6	6.81
La21	15×10	$\mathcal{N}_3^R$	0.88	1.16	1.37		5.05
		$\mathcal{N}_4^R$	0.85	1.07	1.29	64.8	7.66
		$\mathcal{N}_3^R \cup \mathcal{N}_4^R$	0.77	1.06	1.27	41.1	6.99
La24	15×10	$\mathcal{N}_3^R$	0.71	1.24	2.07		4.93
		$\mathcal{N}_4^R$	0.63	1.11	1.49	60.6	7.24
		$\mathcal{N}_3^R \cup \mathcal{N}_4^R$	0.69	1.15	1.50	39.3	6.77
La25	15×10	$\mathcal{N}_3^R$	0.28	0.77	1.19		5.01
		$\mathcal{N}_4^R$	0.27	0.82	1.11	89.7	8.03
		$\mathcal{N}_3^R \cup \mathcal{N}_4^R$	0.26	0.78	1.10	53.2	7.40
La27	20×10	$\mathcal{N}_3^R$	0.89	2.14	2.75		8.94
		$\mathcal{N}_4^R$	0.68	1.77	2.52	107.9	15.69
		$\mathcal{N}_3^R \cup \mathcal{N}_4^R$	0.62	1.71	2.47	61.8	13.79
La29	20×10	$\mathcal{N}_3^R$	1.87	3.47	4.90		8.48
		$\mathcal{N}_4^R$	1.39	2.81	4.06	108.9	14.62
		$\mathcal{N}_3^R \cup \mathcal{N}_4^R$	1.41	2.71	4.21	63.1	12.57
La38	15×15	$\mathcal{N}_3^R$	1.06	2.12	4.16		8.86
		$\mathcal{N}_4^R$	0.98	2.31	3.96	63.2	13.25
		$\mathcal{N}_3^R \cup \mathcal{N}_4^R$	0.95	2.24	4.01	33.3	11.87
La40	15×15	$\mathcal{N}_3^R$	0.82	1.36	2.03		9.17
		$\mathcal{N}_4^R$	0.92	1.38	2.12	67.2	13.93
		$\mathcal{N}_3^R \cup \mathcal{N}_4^R$	0.86	1.32	1.96	34.1	12.64
ABZ7	20×15	$\mathcal{N}_3^R$	2.69	3.93	4.95		15.66
		$\mathcal{N}_4^R$	2.47	3.52	4.54	130.7	26.29
		$\mathcal{N}_3^R \cup \mathcal{N}_4^R$	2.36	3.48	4.52	67.3	22.26
ABZ8	20×15	$\mathcal{N}_3^R$	6.22	7.58	8.89		16.97
		$\mathcal{N}_4^R$	5.81	7.15	8.52	141.1	31.60
		$\mathcal{N}_3^R \cup \mathcal{N}_4^R$	5.65	7.01	8.35	68.0	25.22
ABZ9	20×15	$\mathcal{N}_3^R$	5.54	7.23	8.95		15.92
		$\mathcal{N}_4^R$	4.84	6.61	8.26	103.3	26.62
		$\mathcal{N}_3^R \cup \mathcal{N}_4^R$	4.67	6.51	8.15	51.7	22.56

across each family of ten fuzzy instances of the makespan relative error and of the variation neighbourhood size and the CPU time taken in average by one run. The relative error is calculated w.r.t. the value of the optimal solution of the crisp instance or to a lower bound when the optimal solution is not known.

The results show that $\mathcal{N}_4^R$ breaks the existing quality threshold from [17], improving the relative error of the expected makespan in the best, average and worst solution for almost every instance. In average across all instances the improvement are 7.48%, 10.65% and 9.24% respectively. As expected, the tradeoff is the increase in the number of evaluated neighbours (97.37%) and hence in the CPU time required (66.37%).

Similarly to [13] for the deterministic JSP, both neighbourhoods are combined into an advanced one $\mathcal{N}_3^R \cup \mathcal{N}_4^R$, combining the advantages of both. Such combination may be expected to contain more neighbours and hence require more CPU time. However, it reaches better solutions than $\mathcal{N}_4^R$ evaluating less neighbours. The increase in the number of neighbours evaluated by the MA compared to using $\mathcal{N}_3^R$ is approximately 55%. Additionally, a t-test has been run to compare neighbourhood choices, namely, $\mathcal{N}_3^R$ vs. $\mathcal{N}_4^R$, $\mathcal{N}_3^R$ vs. $\mathcal{N}_3^R \cup \mathcal{N}_4^R$ and $\mathcal{N}_4^R$ vs. $\mathcal{N}_3^R \cup \mathcal{N}_4^R$, using in all cases average makespan values. The results show the existence of statistically significant differences for each choice, with p-value=0.01 in all cases.

There is no clear correlation between instance sizes and makespan results. Instances with 20 jobs, with a large reduction in relative error also have an important increase in the number of neighbours. For square instances of size 15×15, the MA with $\mathcal{N}_3^R$ is better in average than with $\mathcal{N}_4^R$ and sometimes also better than with $\mathcal{N}_3^R \cup \mathcal{N}_4^R$, but this is not the case in all square instances, as we can see for the FT10.

5 Conclusions

We have cosidered a job shop problem with uncertain durations modelled as TFNs. We have proposed a new neighbourhood structure for local search, denoted $\mathcal{N}_4$, based on inserting a critical task into the most extreme position within its block which maintains feasibility. To do this, a sufficient condition for feasibility is provided and the resulting neighbourhood is shown to asymptotically converge to an optimum. A reduced neighbourhood, $\mathcal{N}_4^R$ is obtained by allowing insertion only if it is at the extreme of the block. This allows to reduce the set of neighbours by pruning non-improving moves. Finally, experimental results show the good behaviour of this neighbourhood within a memetic algorithm. The experiments also show that combining $\mathcal{N}_4^R$ with an existing neighbourhood structure from the literature we improve the best results so far whilst considerably reducing neighbourhood size and hence, the CPU time required.

Acknowledgements. All authors are supported by MEC-FEDER Grant TIN2010-20976-C02-02 .

References

1. Applegate, D., Cook, W.: A computational study of the job-shop scheduling problem. ORSA Journal of Computing 3, 149–156 (1991)
2. Brucker, P., Knust, S.: Complex Scheduling. Springer, Heidelberg (2006)

3. Dell' Amico, M., Trubian, M.: Applying tabu search to the job-shop scheduling problem. Annals of Operational Research 41, 231–252 (1993)
4. Dubois, D., Fargier, H., Fortemps, P.: Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge. European Journal of Operational Research 147, 231–252 (2003)
5. Fortemps, P.: Jobshop scheduling with imprecise durations: a fuzzy approach. IEEE Transactions of Fuzzy Systems 7, 557–569 (1997)
6. González Rodríguez, I., Vela, C.R., Puente, J.: Sensitivity Analysis for the Job Shop Problem with Uncertain Durations and Flexible Due Dates. In: Mira, J., Álvarez, J.R. (eds.) IWINAC 2007. LNCS, vol. 4527, pp. 538–547. Springer, Heidelberg (2007)
7. González Rodríguez, I., Puente, J., Vela, C.R., Varela, R.: Semantics of schedules for the fuzzy job shop problem. IEEE Transactions on Systems, Man and Cybernetics, Part A 38(3), 655–666 (2008)
8. González Rodríguez, I., Vela, C.R., Hernández-Arauzo, A., Puente, J.: Improved local search for job shop scheduling with uncertain durations. In: Proc. of ICAPS 2009, pp. 154–161. AAAI Press (2009)
9. González Rodríguez, I., Vela, C.R., Puente, J., Varela, R.: A new local search for the job shop problem with uncertain durations. In: Proc. of ICAPS 2008, pp. 124–131. AAAI Press (2008)
10. Grabowski, J., Wodecki, M.: A very fast tabu search algorithm for job shop problem. In: Metaheuristic Optimization via Memory and Evolution. Tabu Search and Scatter Search. Operations Research/Computer Science Interfaces Series, pp. 117–144. Springer, Heidelberg (2005)
11. Herroelen, W., Leus, R.: Project scheduling under uncertainty: Survey and research potentials. European Journal of Operational Research 165, 289–306 (2005)
12. Liu, B., Liu, Y.K.: Expected value of fuzzy variable and fuzzy expected value models. IEEE Transactions on Fuzzy Systems 10, 445–450 (2002)
13. Mattfeld, D.C.: Evolutionary Search and the Job Shop Investigations on Genetic Algorithms for Production Scheduling. Springer, Heidelberg (1995)
14. Nowicki, E., Smutnicki, C.: A fast taboo search algorithm for the job shop scheduling problem. Management Science 42, 797–813 (1996)
15. Petrovic, S., Fayad, S., Petrovic, D.: Sensitivity analysis of a fuzzy multiobjective scheduling problem. International Journal of Production Research 46(12), 3327–3344 (2007)
16. Pinedo, M.L.: Scheduling. Theory, Algorithms, and Systems, 3rd edn. Springer, Heidelberg (2008)
17. Puente, J., Vela, C.R., González-Rodríguez, I.: Fast local search for fuzzy job shop scheduling. In: Proc. of ECAI 2010, pp. 739–744. IOS Press (2010)
18. Sakawa, M., Kubota, R.: 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)
19. Taillard, E.D.: Parallel taboo search techniques for the job shop scheduling problem. ORSA Journal on Computing 6(2), 108–117 (1994)
20. Tavakkoli-Moghaddam, R., Safei, N., Kah, M.: 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)
21. Van Laarhoven, P., Aarts, E., Lenstra, K.: Job shop scheduling by simulated annealing. Operations Research 40, 113–125 (1992)