Evolutionary tabu search for flexible due-date satisfaction in fuzzy job shop scheduling

Camino R. Vela[a], Sezin Afsar[a], Juan José Palacios[a], Inés González-Rodríguez[b], Jorge Puente[a]

[a]*Department of Computing, University of Oviedo, (Spain)*
[b]*Dept. of Mathematics, Statistics and Computing, University of Cantabria, (Spain)*

Abstract

We consider the job shop scheduling problem with fuzzy sets modelling uncertain durations and flexible due dates. With the goal of maximising due-date satisfaction under uncertainty, we first give a new measure of overall due-date satisfaction in this setting. Then, we define a neighbourhood structure for local search, analyse its theoretical properties and provide a neighbour-estimation procedure. Additionally, a tabu search procedure using the neighbourhood is combined with a genetic algorithm, so the resulting memetic algorithm, guided by the defined due-date satisfaction measure, is run on a set of benchmarks. The obtained results illustrate the potential of our proposal.

Keywords: Scheduling, Fuzzy Sets, Metaheuristics, Due dates, Tabu search
2010 MSC: 90B40, 90B99, 68T20, 68T37

1. Introduction

Scheduling problems are pervasive in a growing number of application domains. One of the most relevant problems in this family is the job shop, since it is considered to be a good model for many practical applications as well as posing a challenge due to its complexity [1].

Even though the most common objective is to find solutions with minimum makespan, due-date satisfaction is receiving increasing attention in recent years [2, 3]. On-time fulfilment emerges as a primary goal in modern pull-oriented supply chain systems and keeping job due dates is a prerequisite for serving customers within the promised delivery time and avoiding out-of-stocks or delay-compensation costs. Hence the importance of considering due-date satisfaction measures that help companies increase their logistic service and create competitive advantage.

Traditionally, it has been assumed that scheduling takes place in static and certain environments. However, for many real-world scheduling problems design variables are subject to perturbations or changes, causing optimal solutions to the original "ideal" problem to be of little or no use in practice. It is also common to handle all constraints as sharp, while in some cases there is certain flexibility and constraints are better expressed in terms of preference, so it is possible to satisfy them to a certain degree.

A source of uncertainty in scheduling is activity durations. Within the great diversity of approaches to this issue, fuzzy sets and possibility theory provide an interesting framework, with a tradeoff between the expressive power of probability and its associated computational complexity and knowledge demands. Fuzzy sets can also be used to model flexibility or gradeness in certain management constraints such as due dates [4]. In fact, in one of the earliest works considering flexible due dates, it is argued that scheduling over long horizons under strict temporal constraints may lead to rejecting an efficient schedule even when the violation of these constraints is insignificant, while in practice, constraints are often relaxable to some extent or subject to preferences [5]. For instance, when due dates are agreed between the manufacturer and the customer, it is contemplated that the product may not be delivered on time subject to penalties for the manufacturer, usually dependent on the tardiness and often proportional to the contract prices [6]. Even in the absence of tardiness penalties, there may be a due time which the customer feels is the most satisfactory delivery time and after which the customer's satisfaction decreases (with unwanted consequences for the manufacturer's reputation). Fuzzy sets provide a suitable and popular framework for representing such flexibility and preferences in temporal constraints, as shown by the abundant references in two recent reviews on fuzzy shop scheduling [7, 8].

In the literature we find some successful applications of fuzzy due dates to real-world problems. Perhaps the earliest example [9] is the building of a course schedule in a French "Grand École" where flexible constraints (such as ending dates for some courses) are modelled using fuzzy sets. In [6], fuzzy due dates are used to model gradual penalisation and decreasing satisfaction degrees for completion times in a Just-In-Time production planning approach for One-of-a-Kind product Manufacturing (OKM) systems, in an approach specifically designed for the computer-integrated manufacturing system of a Blower Works manufacturer in China. A real-life production scheduling problem from an apparel manufacturer is considered in [10] The problem arises in a fabric cutting department

Email addresses: crvela@uniovi.es (Camino R. Vela), afsarsezin@uniovi.es (Sezin Afsar), palaciosjuan@uniovi.es (Juan José Palacios), gonzalezri@unican.es (Inés González-Rodríguez), puente@uniovi.es (Jorge Puente)

feeding sewing assembly lines downstream, where fabric cutting jobs belonging to different production orders have to be processed on one of the parallel spreading tables so that demand from downstream sewing lines can be timely fulfilled. Trapezoidal fuzzy sets are used to model due dates determined by the factory manager for different production orders, representing the managerial preference regarding different values of production order completion time. A genetic algorithm is used to generate just-in-time fabric-cutting schedules. Experimental results on two sets of real production data demonstrate that the genetically optimized schedules improve the internal satisfaction of downstream production departments and reduce the production cost simultaneously. Fuzzy due dates are also used in [11] to represent a decision maker (in this case, a production manager) satisfaction degree with respect to job completion times in a multiobjective single machine scheduling problem. This models a real-world situation arising in a pottery manufacturing company where the single machine is in fact a kiln and meeting due dates is important because further processing of the products is carried out after their firing in the kiln. A genetic algorithm combined with tabu search is applied to find a schedule that maximise two different aggregated due-date satisfaction grades. In [12], the problem under consideration is a job shop scheduling problem which involves batching and lot-sizing in a printing company from the UK that involves 18 machines grouped into 7 work centres. Fuzzy sets are used to represent due dates that are flexible and allow the decision maker to express his/her attitude toward the tardiness of jobs: jobs are classified into three groups according to their priority and the the fuzzy job due dates represent the corresponding tolerance to a delay with respect to the originally set due date. Also, triangular fuzzy numbers are used to model uncertain processing times that stem from staff-operated machines. A fuzzy rule-based system is used to decide on the lot sizing, whereas the fuzzy job shop problem is solved with a genetic algorithm. More recently, [13] tackles a single machine scheduling problem that arises from a cable manufacturing system where cables are produced in different sizes and colours using raw materials of copper and plastic cover on a single machine/system. Fuzzy due dates model increasing penalties associated to the tardiness with respect to an original ideal due date and a greedy algorithm hybridised with variable neighbourhood search and simulated annealing is proposed to solve the resulting method.

The variant of job shop scheduling problem with fuzzy durations and, optionally, fuzzy due dates, is called fuzzy job shop. Most contributions in the literature concentrate on minimising the project's makespan, but some authors have addressed the maximisation of due-date satisfaction, either on its own or in a multiobjective setting, combined with makespan [7].

Metaheuristics are widely recognised as efficient approaches for many hard optimisation problems [14]. Hybrid metaheuristic approaches combining different algorithmic techniques have proved very successful in tackling complex problems of combinatorial nature [15]. Among these, memetic algorithms (MAs) benefit from the interplay between their global and local search components while they try to exploit specific or heuristic knowledge of the problem at hand [16, 17]. Several metaheuristic approaches, including hybrid ones, have been proposed for the fuzzy job shop [7, 18].

In this paper, we intend to advance in the study of the fuzzy job shop scheduling problem, and in particular, in metaheuristic local search methods to maximise due-date satisfaction when uncertain task durations and flexible due dates are fuzzy sets.

1.1. Related work

In the development of solving approaches, local search methods, either on their own or combined with other metaheuristics, provide some of the most competitive solutions to different variants of the job shop problem. In particular, tabu search is a component of most of the state-of-the-art methods for makespan minimisation: it is combined with scatter search for the flexible job shop scheduling problem [19], with a genetic algorithm for the flexible job shop with setup times [20], with a genetic algorithm and heuristic seeding for the fuzzy flexible job shop problem [21] and with simulated annealing for the classical job shop problem [22]. Some theoretical and experimental analysis of the landscape of the solution space for the job shop scheduling problem is presented in [23], trying to shed some light into the success of tabu search for this problem.

The state-of.the-art methods for the deterministic job shop with total weighted tardiness, probably the most used objective related to due-date satisfaction in classical job shop, are also metaheuristics incorporating local search: the Genetic Local Search from [24] which combines a genetic algorithm with an iterated local search, the Hybrid Shifting Bottleneck with Tabu Search approach proposed in [25], the Hybrid Genetic Algorithm with Tabu Search from [26], and the Extended GRASP algorithm presented in [27].

When scheduling takes place in uncertain environments, fuzzy sets, in particular fuzzy numbers, have been used to model uncertain processing times. After the seminal papers [5, 28], many variants of fuzzy job shop have appeared in the literature, and many contributions have been proposed for solving all these variants (cf. the reviews in [7, 8, 18]). The simulated annealing algorithm from [29] and the genetic algorithm from [30] constitute two landmarks in the application of metaheuristic methods to the fuzzy job shop. Currently, some of the most competitive methods for fuzzy makespan minimisation in job shop problems are the genetic algorithm from [31] and the memetic algorithm from [18].

For problems with fuzzy durations and fuzzy due dates, the degree to which a job's fuzzy completion time satisfies the flexible due-date measured as a degree of subsethood (called *agreement index* after [30]) has provided the basis for the most typical objective functions. Maximising the minimum agreement index has been the objective function of different metaheuristics: random-key genetic algorithm [31], scatter search method [32], hybrid discrete imperialist competition algorithm [33] and memetic algorithm [34]. There are also methods aimed at maximising the average agreement index: the memetic algorithm from [34], the co-evolutionary method from [35], here for a fuzzy job shop with multi-process routes, and the multiobjective genetic algorithm from [36], which also attempts to minimise the number of tardy jobs. There are also several multiobjective approaches that attempt to maximise the minimum or the average agreement index together with minimising of the makespan: with a fuzzy decision making approach [37, 38], a lexicographical approach [39] or a Pareto-front approximation approach [40–42].

The state-of-the-art methods for agreement index maximisation depend on the test bed and also on the actual objective function considered. For the classical benchmark from [37] and [30] and minimum agreement index maximisation, the most competitive methods are the hybrid discrete imperialist competition method from [33] and the memetic algorithm from [34]. For the same objective function, on harder instances, the memetic algorithm from [43] outperforms the memetic algorithm from [34] and also provides significantly better results than two multi-meme algorithm hybridised with local search. In [39] and [34] minimum and average agreement index values are reported for another set of instances. In fact, almost full due-date satisfaction is reached with these methods, suggesting there is no room for improvement in this area. However, in [34] and [43] new results on fuzzy instances built from harder classical deterministic job shop instances make it clear that the problem of due-date satisfaction is far from being solved.

2. The fuzzy job shop 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_j, $j=1,\ldots,n$, consists of $m_j \leq m$ tasks $\{\theta_{j1},\ldots,\theta_{jm_j}\}$ to be sequentially scheduled. Also, there are *capacity constraints*, whereby each task θ_{jk} requires the exclusive use of one of the machines with a processing time p_{jk}. Non-preemption and no-recirculation are assumed, meaning that the execution of a task cannot be interrupted and two tasks in the same job cannot require the same machine. Additionally, each job J_j has a due date d_j by which it is desirable that the job be completed. A solution to this problem is a schedule s, i.e. an allocation of starting time s_{jk} for each task θ_{jk}, which is *feasible* (in the sense that all precedence and resource constraints hold) as well as *optimal* according to some criterion, in our case, maximal due-date satisfaction.

2.1. Fuzzy durations and flexible due dates

In real-life applications, it is difficult, if not impossible, to foresee in advance the exact time it will take to process a task. It is reasonable however to have some incomplete knowledge about the duration, possibly based on previous experience. The crudest representation of such uncertain knowledge would be a human-originated confidence interval and, if some values appear to be more plausible than others, then a natural extension is a fuzzy interval or fuzzy number. The simplest model is a *triangular fuzzy number* or *TFN*, denoted $\widehat{a}=(a^1,a^2,a^3)$, given by an interval $[a^1,a^3]$ of possible values and a modal value $a^2\in[a^1,a^3]$, so its membership function takes the following triangular shape, illustrated in Figure 1:

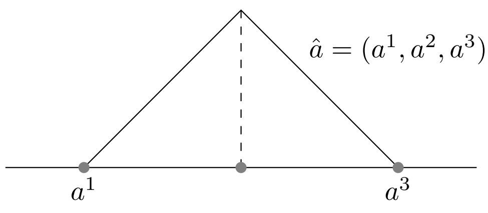


Figure 1: Membership function of a triangular fuzzy number $\widehat{a}$.

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

Triangular fuzzy numbers (or, more generally, fuzzy intervals) are widely used in scheduling as a model for uncertain processing times [4, 7, 18].

The sum and maximum of fuzzy numbers, needed to handle them in the job shop, are usually defined by extending the corresponding operations on real numbers. The resulting addition is pretty straightforward, so for any pair of TFNs $\widehat{a}$ and $\widehat{b}$ we have $\widehat{a}+\widehat{b}=(a^1+b^1,a^2+b^2,a^3+b^3)$. Computing the extended maximum is not that simple and the set of TFNs is not even closed under this operation. Hence, it is common in the fuzzy scheduling literature to approximate the maximum of two TFNs as $\max(\widehat{a},\widehat{b})\approx(\max\{a^1,b^1\},\max\{a^2,b^2\},\max\{a^3,b^3\})$. Besides its extended use, several arguments can be given in favour of this approximation (cf. [18]).

Finally, it is possible to compute the *expected value* of a TFN $\widehat{a}$ as

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

Notice that the expected value is linear with respect to addition and multiplication by a scalar quantity [44].

Regarding due dates, there is often certain flexibility. Consider the case where there is a preferred delivery date d^1, but some delay may be allowed up to a later date d^2. Satisfying the due-date constraint thus becomes a matter of degree. A fuzzy set $\widetilde{d}=(d^1,d^2)$ can be used to model such gradual satisfaction level with a decreasing membership function:

$$\mu_{\widetilde{d}}(x)=\begin{cases}1 & : x\leq d^1\\ \frac{x-d^2}{d^1-d^2} & : d^1<x\leq d^2\\ 0 & : d^2<x\end{cases} \tag{3}$$

This expresses a flexible threshold "less than", representing the satisfaction level $sat(t)=\mu_{\widetilde{d}}(t)$ for the ending date t of the job [4], depicted in Figure 2.

2.2. The disjunctive graph model representation

For the deterministic job shop with makespan minimisation as objective, the disjunctive graph representation provides the basis for many heuristic solving methods [45]. Consequently, variants of the model that cope, among others, with crisp due-date tardiness or lateness and with uncertain durations have

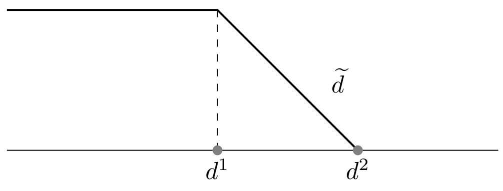


Figure 2: Membership function of a flexible due date $\widetilde{d}$.

been proposed [1–3, 46]. Here we build on these to provide a model for the fuzzy job shop with flexible due dates.

A disjunctive graph is a directed graph $G = (V, A \cup D)$ representing a problem instance. Each node in the set V either represents a task or is a start 0 node or an end node e_j, $1 \leq j \leq n$, corresponding to dummy tasks with null processing times. Arcs in A are called *conjunctive arcs* and represent job-precedence constraints, including directed arcs from node 0 to the first task of each job and arcs from the last task of each job j to the corresponding end node e_j. Arcs in D are called *disjunctive arcs* and represent resource constraints, so D is partitioned into subsets D_k, $D = \cup_{k=1,\dots,m} D_k$, where D_k includes an arc for each pair of tasks requiring machine M_k. All arcs are weighted with the processing time of the task at the source node (a TFN in our case).

Finding a solution to the fuzzy JSP essentially consists in establishing partial task processing orders on all machines, represented by a linear processing order σ or an acyclic subgraph $G(\sigma)$ of G, $G(\sigma) = (V, A \cup R(\sigma))$, where $R(\sigma) = \cup_{k=1\dots m} R_k(\sigma)$, $R_k(\sigma)$ being a hamiltonian selection of D_k. This is often referred to as *solution graph*. A feasible schedule (starting and completion times of all tasks) may be easily computed by simply propagating constraints in $G(\sigma)$ using the sum and maximum of TFNs.

In the deterministic case, critical paths (longest paths from the start node to an end node) play an important role in the design of heuristic methods. Extending the notion of criticality to the problem with fuzzy durations is not trivial, with diverse proposals co-existing in the literature (cf. [4, 29]). Here we follow [46] and, given a solution graph $G(\sigma)$, consider three *parallel solution graphs* $G^i(\sigma)$, $i = 1, 2, 3$, with identical structure to $G(\sigma)$ but where the cost of any arc (x, y) is p^i_x, the i-th component of the processing time $\widehat{p}_x$ for the task at the source node x. Weights in each parallel graph $G^i(\sigma)$ are deterministic, so a critical path in $G^i(\sigma)$ for a job j is the longest path from node 0 to node e_j. The set of *critical paths* in $G(\sigma)$ is defined as the union of critical paths in $G^i(\sigma)$, $i = 1, 2, 3$. *Critical nodes and arcs* are those in a critical path. Unlike the deterministic case, the union set of critical paths from 0 to e_j for all $j = 1, \dots, n$ is not a tree.

Let σ be a task processing order and for every task x with processing time $\widehat{p}_x$, let $PM_x(\sigma)$ and $SM_x(\sigma)$ denote respectively the tasks preceding and succeeding x in the machine sequence provided by σ (in $R(\sigma)$), and let PJ_x and SJ_x denote respectively the predecessor and successor tasks of x in the job sequence (in A). We define the *head* $\widehat{r}_x(\sigma)$ of task x as the starting time of x, a TFN given by:

$$\widehat{r}_x(\sigma) = \max\{\widehat{r}_{PJ_x}(\sigma) + \widehat{p}_{PJ_x}, \widehat{r}_{PM_x(\sigma)}(\sigma) + \widehat{p}_{PM_x(\sigma)}\}, \quad (4)$$

where $\widehat{r}_s(\sigma) = 0$. It corresponds to the length of a longest path in $G(\sigma)$ from 0 to x. We also define the *tail* of task x relative to j, denoted $\widehat{q}_{x,j}(\sigma)$, as:

$$\widehat{q}_{x,j}(\sigma) = \max\{\widehat{q}_{SJ_x,j}(\sigma) + \widehat{p}_{SJ_x}, \widehat{q}_{SM_x(\sigma),j}(\sigma) + \widehat{p}_{SM_x(\sigma)}\}, \quad (5)$$

where $\widehat{q}_{e_j,l}(\sigma) = 0$ if $l = j$ and $-\infty$ otherwise and $\widehat{q}_{SM_x(\sigma),j}(\sigma) = -\infty$ if x is the last task to be processed in its machine. The tail represents the time left after x until a job is completed, i.e., the length of a longest path in $G(\sigma)$ from the successors of x to the job's end node. Notice that the tail of a task x w.r.t. j needs to be computed even if x does not belong to job J_j and it is trivial, $-\infty$, when there is no path from x to e_j. When there is no confusion regarding the processing order, we may simply write $\widehat{r}_x$ and $\widehat{q}_{x,j}$.

Let $\widehat{c}_x(\sigma)$ and $\widehat{c}_j(\sigma)$ denote respectively the completion times of a task x and job j (or simply $\widehat{c}_x$ and $\widehat{c}_j$). Clearly, $\widehat{c}_x = \widehat{r}_x + \widehat{p}_x$ and $\widehat{c}_j = \widehat{r}_{e_j}$. For each parallel graph $G^i(\sigma)$, r^i_x is the length of the longest path from node 0 to node x and $q^i_{x,j} + p^i_x$ is the length of the longest path from node x to node e_j if this path exists. Hence, for all i, $r^i_x + p^i_{x,j} + q^i_x$ is the length of the longest path from node 0 to node e_j through node x in $G^i(\sigma)$; it is a lower bound of the i-th component of $\widehat{c}_j$, being equal to c^i_j if node x belongs to a critical path in $G^i(\sigma)$ for J_j. The following properties ensue trivially from the definition.

Proposition 1. *If an arc $v = (x, y)$ is critical for some job J_j, then $\exists i$ such that $r^i_x + p^i_x = r^i_y$. A task x is critical for some job J_j if and only if $\exists i$ such that $r^i_x + p^i_x + q^i_{x,j} = c^i_j$.*

2.3. Due-date satisfaction for fuzzy schedules

The schedule obtained from a task processing order σ is fuzzy in the sense that the starting and completion times of all tasks are TFNs, interpreted as possibility distributions on the values that the times may take. In particular, every job's completion time is no longer a real number but a TFN $\widehat{c}$, so the degree to which $\widehat{c}$ satisfies a due-date constraint $\widetilde{d}$ may no longer be measured by direct evaluation of $\mu_{\widetilde{d}}$.

The most common approach to this problem is to use the so-called *agreement index* or *AI* [7], which essentially corresponds to the degree of subsethood of the completion time into the due date. However, treating both the due date and the completion time equally can be objected to since they have very different meanings, with one of them modelling uncertainty while the other one models flexibility. Also, fuzzy schedules can be seen as predictive or a-priori schedules, found when the duration of tasks is not exactly known and a set of possible scenarios must be taken into account [38]. At this stage, the degree of due-date satisfaction can only be approximated; only after tasks have been executed according to the ordering provided by the schedule and real durations are known can the actual due-date satisfaction degree be measured. The agreement index thus corresponds to an estimate of the due-date satisfaction degree that would be obtained after execution, but it tends to be excessively optimistic (and innacurate) since it ignores the information provided by the portion of $\widehat{c}$ outside the support of the due date $\widetilde{d}$.

We propose instead an alternative measure that takes into account the different nature of fuzzy completion times and fuzzy due dates and also considers all possible realisations of the job completion time.

Definition 1. Let $\widehat{c}$ be a fuzzy job completion time and let $\widetilde{d}$ be a flexible due date. The *expected satisfaction degree (ESD)* of $\widetilde{d}$ by $\widehat{c}$ is defined as

$$ESD(\widehat{c},\widetilde{d}) = \mu_{\widetilde{d}}(E[\widehat{c}]). \tag{6}$$

The ESD corresponds to the degree to which the expected completion time $E[\widehat{c}]$ satisfies the due date $\widetilde{d}$. This estimate ranges between 0, when the due date is not expected to be satisfied at all, and 1, when it is expected that the due date is fully satisfied. It is illustrated in Figure 3. In the case that the completion time is crisp (absence of uncertainty), ESD becomes the usual measure of flexible due-date satisfaction given by the membership function.

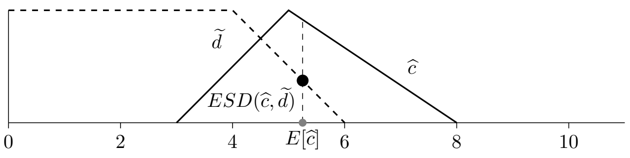


Figure 3: $ESD(\widehat{c},\widetilde{d})$.

Having built a schedule from σ, the expected satisfaction degree $ESD(\widehat{c}_j(\sigma),\widetilde{d}_j)$, denoted ESD_j for short, measures to what degree the job's flexible due date $\widetilde{d}_j$ is satisfied in this schedule, $j = 1,\dots,n$. The overall value of due-date satisfaction for the schedule can be obtained as the average expected satisfaction degree

$$ESD_{avg}(\sigma) = \frac{1}{n}\sum_{j=1,\dots,n} ESD_j, \tag{7}$$

a value that needs to be maximised. Accordingly, an optimal task ordering is one that yields a schedule with maximal value of ESD_{avg}.

Definition 2. Let Σ be the set of feasible task processing orders. A task processing order $\sigma_0 \in \Sigma$ is said to be *optimal* for ESD_{avg} if and only if $ESD_{avg}(\sigma_0) = \max\{ESD_{avg}(\sigma) : \sigma \in \Sigma\}$.

The resulting job shop problem, with fuzzy processing times and fuzzy due dates, and where the objective is to maximise the aggregated expected satisfaction degree ESD_{avg} can be denoted $J|\widehat{p}_j,\widetilde{d}_j|ESD_{avg}$ according to the three-field notation from [47].

2.4. Example

To illustrate the FJSP with due date satisfaction, consider a problem of $n = 3$ jobs and $m = 2$ machines with the following matrices for fuzzy processing times, machine allocation and fuzzy due dates:

$$\widehat{p} = \begin{pmatrix} (3,4,7) & (1,2,3) \\ (4,5,6) & (2,3,4) \\ (1,2,6) & (1,2,5) \end{pmatrix} \nu = \begin{pmatrix} 1 & 2 \\ 2 & 1 \\ 2 & 1 \end{pmatrix} \widetilde{d} = \begin{pmatrix} (4,6) \\ (10,12) \\ (10,12) \end{pmatrix}$$

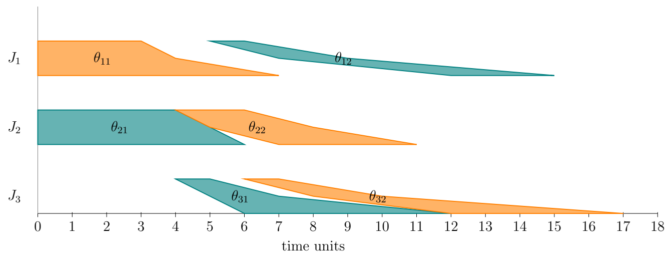


Figure 4: Gantt chart of the schedule represented by the task linear order $\theta_{11}\ \theta_{21}\ \theta_{31}\ \theta_{22}\ \theta_{12}\ \theta_{32}$

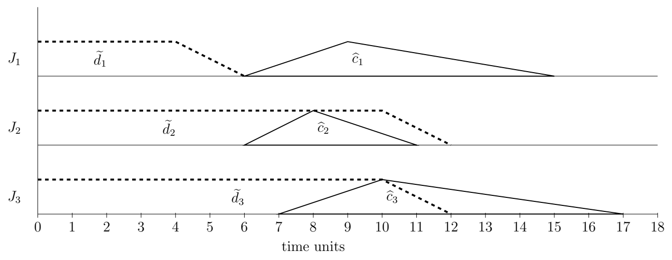


Figure 5: Due dates of jobs and completion times for example of Figure 4

Figure 4 shows the Gantt chart (adapted to TFNs following [29]) of the schedule given by the task processing ordering $\sigma = (\theta_{11}\ \theta_{21}\ \theta_{31}\ \theta_{22}\ \theta_{12}\ \theta_{32})$, using different colours for tasks in different machines. It depicts the scheduled starting times of tasks in each job each job computed following (4): for J_1, $\widehat{s}_{11} = (0,0,0)$, $\widehat{s}_{12} = (5,7,12)$; for J_2, $\widehat{s}_{21} = (0,0,0)$, $\widehat{s}_{22} = (4,5,7)$, and for J_3, $\widehat{s}_{31} = (4,5,6)$, $\widehat{s}_{32} = (6,8,12)$. Figure 5 shows the completion times of jobs $\widehat{c}_1 = (6,9,15)$, $\widehat{c}_2 = (6,8,11)$ and $\widehat{c}_1 = (7,10,17)$ and the corresponding due-dates. It turns out that $ESD_1 = 0$, $ESD_2 = 1$ and $ESD_3 = 0.5$, so, $ESD_{avg} = 0.5$.

The solution graph for this example can be seen in Figure 6.

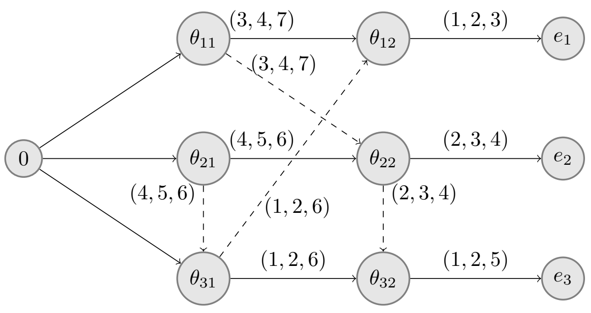
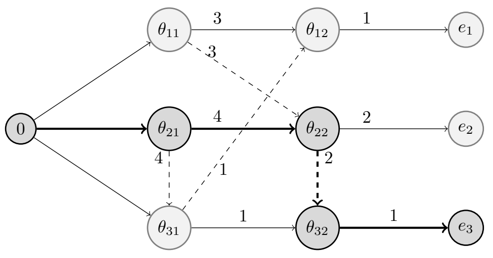
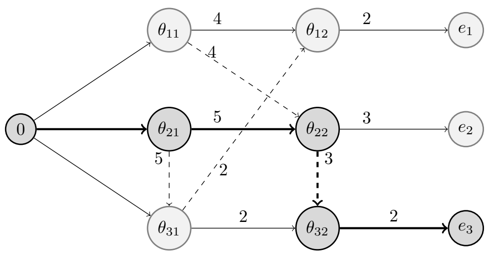


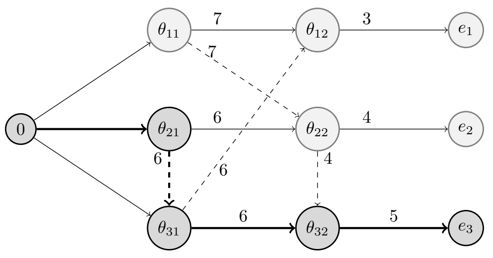
Figure 6: Solution graph $G(\sigma)$ for the processing order $\sigma = (\theta_{11}\ \theta_{21}\ \theta_{31}\ \theta_{22}\ \theta_{12}\ \theta_{32})$ corresponding to the Gantt chart of Figure 4. $c_1(\sigma) = (6,9,15)$, $c_2(\sigma) = (6,8,11)$ and $c_3(\sigma) = (7,10,17)$



G^1. Critical path for job $J_3 = (s\,\theta_{21}\,\theta_{22}\,\theta_{32}\,e_3)$. $C_3^1 = 7$



G^2. Critical path for job $J_3 = (s\,\theta_{21}\,\theta_{22}\,\theta_{32}\,e_3)$. $C_3^2 = 10$



G^3. Critical path for job $J_3 = (s\,\theta_{21}\,\theta_{31}\,\theta_{32}\,e_3)$. $C_3^3 = 17$

Figure 7: Parallel graphs corresponding to the graph in Figure 6, with critical paths in red.

Finally, Figure 7 shows the parallel solutions graphs corresponding to the solution graph of figure 6 with the critical paths to job J_3 highlighted; it is easy to check that the set of those critical paths is not a tree in this example.

2.5. *MILP model*

We finish this section on the fuzzy job shop problem with flexible due date satisfaction by modelling it as a mixed integer linear program. This will allow us to use a commercial solver, IBM ILOG CPLEX, to try to solve it.

Let J denote the set of job indices, that is, $J = \{1, \ldots, n\}$, let M denote the set of machine indices, $M = \{1, \ldots, m\}$, let M_j denote the set of indices of the machines required to process the tasks of job J_j, let n_l denote the number of tasks that need to be processed in the l-th machine, $l \in M$, and let W be a sufficiently large number. Then, a MILP formulation for the fuzzy job shop with flexible due date satisfaction can be given as follows:

(P1)

$$\max \frac{1}{n} \sum_{j \in J} ESD_j \tag{8}$$

s.t.

$$\sum_{\substack{i,j \in J \\ i \neq j}} x_{ijl} = n_l - 1 \qquad \forall l \in M_j \cap M_i \tag{9}$$

$$\sum_{\substack{i \in J \\ i \neq j}} x_{ijl} \leq 1 \qquad \forall j \in J, l \in M_j \cap M_i \tag{10}$$

$$\sum_{\substack{j \in J \\ j \neq i}} x_{ijl} \leq 1 \qquad \forall i \in J, l \in M_j \cap M_i \tag{11}$$

$$s_{jl}^k \geq s_{il}^k + p_{il}^k - W \cdot (1 - x_{ijl}) \qquad \forall k \in \{1,2,3\}, \; i,j \in J, l \in M_j \cap M_i \tag{12}$$

$$s_\theta^k \geq s_{PJ_\theta}^k + p_{PJ_\theta}^k \qquad \forall k \in \{1,2,3\}, \theta \in \Theta \tag{13}$$

$$0 \leq s_\theta^1 \leq s_\theta^2 \leq s_\theta^3 \qquad \forall \theta \in \Theta \tag{14}$$

$$x_{ijl} \in \{0,1\} \qquad \forall i,j \in J, l \in M_j \cap M_i \tag{15}$$

Here, the binary decision variable x_{ijl} takes value 1 if job i immediately precedes job j on machine l and is 0 otherwise. Note that x_{ijl} is defined if a task of job j and a task of job i are processed on machine l, i.e., if $l \in M_j \cap M_i$. Constraint (9) ensures that exactly n_l tasks are processed in each machine $l \in M_j \cap M_i$, while constraints (10) and (11) ensure that each task has at most one immediate predecessor and one immediate successor in its machine. According to constraint (12), there is no overlap in the processing of two consecutive tasks in a machine. Lastly, constraint (13) enforces the right task order within jobs and (14) ensures that the three components s_x^1, s_x^2, s_x^3 of each task starting time actually define a TFN $\widehat{s}_x = (s_x^1, s_x^2, s_x^3)$.

3. Tabu search to maximise ESD_{avg}

Generally speaking, a local search starts from a given solution and at each step it selects a promising element from a neighbourhood structure to replace the current solution, until a stopping criterion is met. In tabu search, the effectiveness and efficiency of the exploration is improved by storing some historical information of the search in a so-called tabu list.

Clearly, a key point of any local search is the definition of a neighbourhood structure. It is also important to have an efficient means of selecting promising neighbours, in order to keep the computational cost of the local search within reasonable bounds. The next subsections describe, respectively, a new neighbourhood structure for *FJSP* to maximise ESD_{avg} (including some properties thereof) and a procedure for ESD_{avg} estimation. Finally, a third subsection describes the evolutionary tabu search algorithm that uses the neighbourhood structure and the estimation procedure to solve the job shop problem.

3.1. *Neighbourhood structure*

For the classical job shop, a well-known neighbourhood, which relies on the concept of critical path is that proposed in [48], later extended to the fuzzy case in [46]. In this structure, neighbours are obtained by reversing single critical arcs.

Based on these, we propose a new neighbourhood structure for ESD_{avg} maximisation. The intuition is that, for ESD_{avg} to increase in a neighbouring solution, it must be the case that at least one of the expected satisfaction degree values ESD_j improves. This occurs only if the due date d_j is not yet fully satisfied and the completion time $\widehat{c}_j(\sigma)$ of the corresponding job is reduced or, equivalently, the length of the longest path from the start node 0 to the end node e_j in the solution graph is reduced.

Let $CP_{\sigma,j}$ denote the set of critical paths for job $J_j, j = 1,\dots,n$ in the solution graph $G(\sigma)$. An improvement in $\widehat{c}_j(\sigma)$ (and hence in ESD_{avg}) can only be obtained by reversing machine arcs belonging to paths in $CP_{\sigma,j}$ for some $j = 1,\dots,n$. Furthermore, since $ESD_j \leq 1$ for $j = 1,\dots,n$, reducing the completion time of a job such that $ESD_j = 1$ (i.e., its due date $\widetilde{d}_j$ is already fully satisfied) cannot improve ESD_{avg} either.

Proposition 2. *Let σ be a feasible processing order, let v be an arc that is not critical for any job J_j such that $ESD_j(\sigma) < 1$ in $G(\sigma)$ and let $\sigma_{(v)}$ denote the processing order obtained from σ after reversing arc v in $G(\sigma)$. Then*

$$\forall j, \quad ESD_j(\sigma_{(v)}) \leq ESD_j(\sigma). \tag{16}$$

PROOF. Let J_j be a job such that $ESD_j(\sigma) < 1$ (if $ESD_j(\sigma) = 1$, it has reached its upper bound and the result follows trivially). If $v = (x,y)$ is not critical for j then, for all $i \in 1,2,3$, the longest paths from 0 to e_j in $G^i(\sigma)$ remain paths in $G^i(\sigma_{(v)})$ too, so $\forall i$ $c^i_j(\sigma_{(v)}) \geq c^i_j(\sigma)$. Therefore, $E[c_j(\sigma_{(v)})] \geq E[c_j(\sigma)]$ and, since $\mu_{\widetilde{d}}$ is a decreasing function, $ESD_j(\sigma_{(v)}) \leq ESD_j(\sigma)$.

This suggests the following definition:

Definition 3. (Neighbourhood $\mathcal{N}_{ESD}$) Let $\sigma \in \Sigma$ be a feasible operation processing order. The *neighbourhood structure* obtained from σ is given by $\mathcal{N}_{ESD}(\sigma) = \{\sigma_{(v)} : v \in R(\sigma)$ is critical for some $1 \leq j \leq n$ with $ESD_j(\sigma) < 1\}$.

The new neighbourhood $\mathcal{N}_{ESD}$ 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, all neighbours in $\mathcal{N}_{ESD}(\sigma)$ are feasible, that is, $\mathcal{N}_{ESD}(\sigma) \subset \Sigma$.*

PROOF. If $v = (x,y)$ is critical, by Proposition 1, $\exists i, r^i_y = r^i_x + p^i_x$. Suppose by contradiction that $G(\sigma_{(v)})$ has a cycle. Since $G(\sigma)$ has no cycles and the only change in $\sigma_{(v)}$ w.r.t. σ has been the processing order between x and y, having a cycle in $G(\sigma_{(v)})$ means that there must exist an alternative path from x to y in $G(\sigma)$. Given the definition of the head $\widehat{r}_y$ in (4), $\forall i$ we have $r^i_y \geq r^i_{PJ_y} + p^i_{PJ_y}$. This, together with the existence of an alternative path from x to y means that

$$\forall i, r^i_y \geq r^i_{SJ_x} + p^i_{SJ_x} + p^i_{PJ_y} \geq r^i_x + p^i_x + p^i_{SJ_x} + p^i_{PJ_y}$$

But being critical, there is at least one component k where $r^k_y = r^k_x + p^k_x$, and for this k we have $r^k_x + p^k_x \geq r^k_x + p^k_x + p^k_{SJ_x} + p^k_{PJ_y}$, which is absurd, because all task durations are strictly positive.

Feasibility means that the local search is automatically limited to the subspace of feasible task orders, thus avoiding feasibility checks and repairing strategies for neighbours. This increases the efficiency of the local search procedure and avoids the loss of feasible solutions that is usually encountered with feasibility checking procedures (cf. [49]).

To prove connectivity, we require two preliminary results.

Lemma 1. *Let $\sigma \in \Sigma$ be a feasible task processing order and let $G(\sigma) = (V, A \cup R(\sigma))$ be its solution graph. If σ is not optimal for ESD_{avg}, then there exists at least one arc in $R(\sigma)$ that is critical for some j with $ESD_j < 1$.*

PROOF. By contradiction, suppose there are no critical arcs for any job in $R(\sigma)$. This means that all critical arcs for any j in $G(\sigma)$ belong to A. Therefore, for all i, every critical path in $G^i(\sigma)$ for any j belongs to A, that is, in each $G^i(\sigma)$ there exists a critical path to each end node e_j where all arcs belong to A. Clearly, the length of such path in $G^i(\sigma)$, $c^i_j(\sigma)$, provides a lower bound for the i-th component of the job's completion time for any other solution π: $\forall i, c^i_j(\sigma) \leq c^i_j(\pi)$. Consequently, $ESD_j(\sigma) \geq ESD_j(\pi), \forall \pi$. Since this is the case for every j, $\forall \pi \in \Sigma, ESD_{avg}(\sigma) \geq ESD_{avg}(\pi)$. Therefore, σ is optimal for ESD_{avg}.

Suppose instead that $R(\sigma)$ contains critical arcs but for every arc $v \in R(\sigma)$ that is critical for some j it holds that $ESD_j(\sigma) = 1$. Then, since this is an upper bound of ESD, it also holds that σ is optimal for ESD_{avg}.

Lemma 2. *Given $\sigma \in \Sigma$ a feasible task processing order, $G(\sigma) = (V, A \cup R(\sigma))$ its solution graph and σ_0 an optimal processing order, let*

$$V_{ESD}(\sigma,\sigma_0) = \{v = (x,y) \in R(\sigma) : v \text{ is critical}$$
$$\text{for some } j \text{ with } ESD_j < 1, (y,x) \in \overline{R(\sigma_0)}\} \tag{17}$$

where $\overline{R(\sigma_0)}$ denotes the transitive closure of $R(\sigma_0)$; i.e., $V_{ESD}(\sigma,\sigma_0)$ is the set of critical arcs (x,y) in $\mathcal{N}_{ESD}(\sigma)$ such that there exists a path from y to x in $R(\sigma_0)$. If $V_{ESD}(\sigma,\sigma_0) = \emptyset$ then σ is optimal w.r.t. ESD_{avg}.

PROOF. From Lemma 1, we know that if σ is not optimal for ESD_{avg} there exists at least one arc in $R(\sigma)$ critical for some j such that $ESD_j < 1$.

We now prove that if σ is not optimal for ESD_{avg}, there exists at least a critical arc for some j with $ESD_j(\sigma) < 1$ such that $(y,x) \in \overline{R(\sigma_0)}$.

Indeed, let us suppose that for every job J_j such that $ESD_j < 1$ every critical arc for j, $v = (x,y) \in R(\sigma)$ verifies that $(x,y) \in \overline{R(\sigma_0)}$ (i.e. $V_{ESD}(\sigma,\sigma_0) = \emptyset$, or equivalently, all critical arcs belong to the transitive closure $\overline{R(\sigma_0)}$). For every j, the set of critical arcs for j in $G(\sigma)$ is the union of the set of critical arcs for j across all parallel disjunctive graphs. Therefore, the assumption means that for all j and for all i, all critical arcs for j in $R^i(\sigma)$ belong to the transitive closure $\overline{R^i(\sigma_0)}$. Hence, a critical path to e_j such that $ESD_j < 1$, P^i_j, in $G^i(\sigma)$ is also critical for j in $\overline{G^i(\sigma_0)} = (V, A \cup \overline{R^i(\sigma_0)})$. Let Q^i_j denote an arbitrary critical path in $\overline{G^i(\sigma_0)}$ ending in e_j and let $\|Q^i_j\|$ denote its length. Since P^i_j is critical for j in $\overline{G^i(\sigma_0)}$, it holds $\forall j, \forall i$, $c^i_j(\sigma) = \|P^i_j\| \leq \|Q^i_j\| = c^i_j(\sigma_0)$, that is, $\forall j, ESD_j(\sigma) \geq ESD_j(\sigma_0)$, so, $ESD_{avg}(\sigma) \geq ESD_{avg}(\sigma_0)$. But, since σ_0 is optimal, $ESD_{avg}(\sigma) \leq ESD_{avg}(\sigma_0)$. Both inequalities mean that $ESD_{avg}(\sigma) = ESD_{avg}(\sigma_0)$ and, hence, σ is optimal w.r.t. ESD_{avg}.

Theorem 2. *$\mathcal{N}_{ESD}$ verifies the connectivity property: for every non-optimal task processing order σ we may build a finite sequence of transitions of $\mathcal{N}_{ESD}$ leading from σ to a globally optimal processing order σ_0.*

PROOF. Let σ_0 be any optimal processing order and let the sequence $\{\lambda_k\}_{k\geq 0}$ of processing orders be given by $\lambda_0 = \sigma$ and $\lambda_{k+1} = \lambda_{k(v)}$ with $v \in V_{ESD}(\lambda_k, \sigma_0)$. The reversal of an arc in $V_{ESD}(\lambda_k, \sigma_0)$ is a move from $\mathcal{N}_{ESD}$ so, by Theorem 1, $\forall k\ \lambda_k \in \Sigma$ (i.e., all task processing orders in the sequence are feasible solutions). Let us prove that the above sequence is finite. For any processing order $\sigma \in \Sigma$, we define

$$M(\sigma, \sigma_0) = \{v = (x,y) \in R(\sigma) : (y,x) \in \overline{R(\sigma_0)}\}$$
$$\overline{M(\sigma, \sigma_0)} = \{v = (x,y) \in \overline{R(\sigma)} : (y,x) \in \overline{R(\sigma_0)}\}$$

Clearly, $V_{ESD}(\lambda_k, \sigma_0) \subset M(\sigma, \sigma_0) \subset \overline{M(\sigma, \sigma_0)}$. Let $\|M(\sigma, \sigma_0)\|$ and $\|\overline{M(\sigma, \sigma_0)}\|$ denote their cardinalities. By definition of λ_k, if $\|M(\lambda_k, \sigma_0)\| > 0$ then $\|\overline{M(\lambda_{k+1}, \sigma_0)}\| = \|\overline{M(\lambda_k, \sigma_0)}\| - 1$. Therefore, for $k^\star = \|\overline{M(\sigma, \sigma_0)}\|$, we have $\|M(\lambda_{k^\star}, \sigma_0)\| = 0$. Since $V_{ESD}(\sigma, \sigma_0) \subseteq M(\sigma, \sigma_0) \subseteq \overline{M(\sigma, \sigma_0)}$, this implies that $V_{ESD}(\lambda_{k^\star}, \sigma_0) = \emptyset$ so, by Lemma 2, $\lambda_{k^\star}$ is optimal for ESD_{avg}.

Connectivity is important because it ensures the non-existence of starting points from which a local search procedure using the neighbourhood structure cannot reach a global optimum. It also ensures asymptotic convergence in probability to a global optimum. Additionally, it opens the possibility of designing an exact branch and bound method for fuzzy job shop.

3.2. ESD estimation procedure

For a feasible processing order σ, evaluating all the neighbours in $\mathcal{N}_{ESD}(\sigma)$ may have a high computational cost. In the worst case where all arcs in $R(\sigma)$ are critical, the number of neighbours $|\mathcal{N}_{ESD}(\sigma)|$ is $|R(\sigma)| = \sum_{k=1}^{m} |R_k(\sigma)|$. In a FJSP problem without reentrant jobs, the greatest possible value for $|R_k(\sigma)|$ is n; therefore, $\mathcal{N}_{ESD}(\sigma)$ contains at most nm neighbours. In addition, evaluating a neighbour $\sigma_{(v)}$ obtained after reversing an arc v in σ in the worst case means recalculating the starting times of all tasks, with a computational cost in $O(nm)$. Hence the computational cost of evaluating all neighbours of a solution is $O(n^2m^2)$. However this cost can be reduced if an upper bound of ESD_{avg} can be easily evaluated that allows to discard non-improving neighbours.

For a processing order σ and tasks x and y, let $P_{\sigma,j}(x \vee y)$ denote the set of all paths to end node e_j in $G(\sigma)$ containing x or y, $P_{\sigma,j}(x \wedge y)$ denote the set of all paths to e_j in $G(\sigma)$ containing both x and y and let $P_{\sigma,j}(\neg x)$ denote de set of all paths to e_j in $G(\sigma)$ not containing x. Also, for a given set P_j of paths to node e_j, let $D[P_j]$ denote the TFN such that $D^i[P_j]$ is given by the length of the longest path in P_j in the parallel graph G^i if $P_j \neq \emptyset$ and $-\infty$ otherwise, $i = 1,2,3$.

Proposition 3. *Let σ be a task processing order and let $\sigma_{(v)}$ be the order that results after reversing $v = (x,y)$ an arc in $G(\sigma)$. Then, the completion time for job j in the new solution is given by:*

$$c_j(\sigma_{(v)}) = \max\{D[P_{\sigma_{(v)},j}(x \vee y)], D[P_{\sigma,j}(\neg x)]\} \tag{18}$$

PROOF. For every $i = 1,2,3$ it holds that $c^i_j(\sigma_{(v)}) = \max\{D^i[P_{\sigma_{(v)},j}(x \vee y)], D^i[P_{\sigma_{(v)},j}(\neg x \wedge \neg y)]\}$. Since the only arcs that change from $G(\sigma)$ to $G(\sigma_{(v)})$ are $(PM_x(\sigma), x)$, (x,y), $(y, SM_y(\sigma))$, those paths to e_j not containing x nor y remain unchanged, i.e., $c^i_j(\sigma_{(v)}) = \max\{D^i[P_{\sigma_{(v)},j}(x \vee y)], D^i[P_{\sigma,j}(\neg x \wedge \neg y)]\}$. Now, for every path to e_j in $G(\sigma)$ containing y but not containing x, either it starts in y or it contains (PJ_y, y) so in any case the subpath to y is identical in both $G(\sigma)$ and $G(\sigma_{(v)})$. If the path to e_j does not contain SM_y, it is still a path to e_j in $G(\sigma_{(v)})$ and if it does contain the arc (y, SM_y), then substituting (x,y) by $(y,x), (x, SM_y)$ we obtain a longer path to e_j in $G(\sigma_{(v)})$. Therefore, $D^i[P_{\sigma,j}(\neg x \wedge y)] \leq D^i[P_{\sigma_{(v)},j}(x \vee y)]$ and we may rewrite the above expression as $c^i_j(\sigma_{(v)}) = \max\{D^i[P_{\sigma_{(v)},j}(x \vee y)], D^i[P_{\sigma,j}(\neg x)]\}$.

Corollary 3. *Let $\widehat{r}'_x$ and $\widehat{r}'_y$ denote respectively the heads of x and y and let $\widehat{q}'_{x,j}$ and $\widehat{q}'_{y,j}$ denote the tails of x and y relative to j after reversing arc $v = (x,y)$ in $G(\sigma)$. Then,*

$$LB(\widehat{c}_j) = \begin{cases} \max\{\widehat{r}'_x + \widehat{p}_x + \widehat{q}'_{x,j}, \widehat{r}'_y + \widehat{p}_y + \widehat{q}'_{y,j}\} \\ \qquad\qquad\qquad \text{if } P_{\sigma_{(v)},j}(x \vee y) \neq \emptyset \\ \widehat{r}_{e_j}(\sigma) \quad \text{otherwise} \end{cases} \tag{19}$$

provides a lower bound *for the completion time c_j of job j obtained after reversing (x,y) in the sense that for $i = 1,2,3$ $LB(\widehat{c}_j)^i \leq c^i_j$ and hence $E[LB(\widehat{c}_j)] \leq E[\widehat{c}_j]$). Consequently,*

$$UB_j = ESD(LB(\widehat{c}_j), \widetilde{d}_j) \tag{20}$$

provides an upper bound *for the expected satisfaction degree for job j and*

$$UB = \frac{1}{n} \sum_{j=1,\dots,n} UB_j \tag{21}$$

provides an upper bound *for the average expected satisfaction degree ESD_{avg}.*

This upper bound of ESD_{avg}, inspired by the work of [50] for the classical job shop with makespan minimisation, is calculated in time $O(n)$ provided that heads and tails of all tasks are known. Notice however that for UB to be computed, not only the starting time of each task must be known, but also its tails relative to the n jobs. This means that, when evaluating a neighbour, besides updating the starting times of all tasks it is also necessary to update the tails of each of the nm tasks, with an additional computational cost of $O(n)$, so if UB is to be calculated, the overall the computational cost of evaluating a neighbour increases to $O(n^2m)$.

3.3. A hybrid evolutionary tabu search method

Tabu search (TS), originally proposed in [51], has been successfully and extensively applied to several optimisation problems since then [52]. A key feature of tabu search is the use of a short-memory structure called tabu list to allow the local search to overcome local optima. The basic principle is to continue the local search after encountering a local optimum by allowing non-improving moves. The reversal or repetition of certain previous moves is avoided by storing selected attributes of recent moves as forbidden or tabu. Thus, the search does not enter a cycling behaviour and follows new trajectories in the search space. In our case, the tabu list structure and its updating strategy follows [49]: for a given move it will store the arc that has been reversed and it will behave as a FIFO list with dynamic length, incorporating cycle-detection. There is also a long-term memory mechanism that stores the best solution found so far.

Tabu restrictions are counterbalanced by the aspiration criterion. In principle, a neighbour is tabu if it is generated by reversing a tabu arc, but its tabu status is lifted if it is better than the best neighbour found so far. The search stops after a maximum number of iterations without improvement or if a cycle is detected.

A pseudocode description of the tabu search method used herein can be seen in Algorithm 1. It differs from the standard TS template in that it incorporates a filtering mechanism to lighten the computational load of neighbour evaluations, as explained below.

3.3.1. Filtering mechanism for neighbour evaluation

At each iteration, a new solution needs to be selected from the set of non-tabu neighbours together with those tabu neighbours satisfying the aspiration criterion. This selection may be based on neighbours' quality estimates rather than on the exact objective function values, in order to avoid the computational burden of evaluating all neighbours [26, 27, 50]. However, while estimates based on Taillard's lower bound such as the one proposed in Section 3.2 provide good surrogates of the actual makespan, several authors have argued that this is not the case for due-date related measures [26, 27].

Here we propose an alternative approach, combining estimates as a filtering mechanism to avoid unnecessary evaluations with exact evaluations of the filtered neighbours to select the best one. For any neighbour s' and associated processing order σ, the estimate of $ESD_{avg}(\sigma)$ is taken to be the value $UB(\sigma)$ from equation (21) in Corollary 3. We abuse notation slightly and write $UB(s')$ and $ESD_{avg}(s')$, identifying the solution with the induced task processing order. Then, the neighbour selection mechanism is as follows. Let $s'_1,\dots,s'_k$ be the generated neighbours in decreasing order of the estimate, that is $UB(s'_1) \geq UB(s'_2) \geq \dots \geq UB(s'_k)$. Starting with s'_1, the selection procedure evaluates neighbours following this order until a neighbour k^* is found such that $max_{1\leq l\leq k^*}\{ESD_{avg}(s'_l)\} \geq UB(s'_{k^*+1})$ (or until all neighbours have been evaluated). The best neighbour is selected from the subset formed by the first k^* neighbours $\{s'_1,\dots,s'_{k^*}\}$, while the remaining $k-k^*$ neighbours $\{s'_{k^*+1},\dots,s'_k\}$ are discarded without evaluating them. Notice that $UB(s'_{k^*+1})$ is an upper bound of $ESD_{avg}(s'_{k^*+1})$ and $UB(s'_{k^*+1}) \geq UB(s'_t)$ for the remaining neighbours $k^*+1 \leq t \leq k$, so it is also the case that $max_{1\leq l\leq k^*}\{ESD_{avg}(s'_l)\} \geq UB(s'_{k^*+1}) \geq ESD_{avg}(s'_t)$ for all remaining neighbours $k^*+1 \leq t \leq k$. In consequence, the best neighbour according to ESD_{avg} must be in the filtered set of $k^* \leq k$ neighbours.

If the filtering mechanism is used, the upper bound $UB(s'_l)$ needs to be computed for all neighbours $s'_l, l = 1,\dots,k$, while a full evaluation of ESD_{avg} is required for the first $k^* \leq k$ neighbours. The computational cost of the filtering mechanism will then depend on k^*. Let us recall from Section 3.2 that, when the filtering mechanism is used, the cost of evaluating a neighbour is $O(n^2m)$ and the cost of computing UB is $O(n)$. Therefore, in the case where $k^* = 1$ and only one neighbour needs to be fully evaluated, the cost of using the filtering mechanism will be in $O(n^2m) + O(n^2m) = O(n^2m)$, comparing favourably with simply evaluating all neighbours without filter which had a cost in $O(n^2m^2)$. However, in the case where $k^* = nm$ and all neighbours need to be fully evaluated after computing the upper bound, the overall cost of the filtering mechanism goes up to $O(n^3m^2)$. We conclude then that the computational cost of using the filtering mechanism and its potential benefit depends heavily on the value k^*. We will assess empirically the actual impact of incorporating the filtering mechanism in Section 4.

3.3.2. Evolutionary tabu search

A problem encountered by all local search (single-solution or trajectory-based) methods is that they tend to restrict themselves to a portion of the search of space, usually dependent on the initial solution provided. This is also the case of tabu search, despite the greater diversification obtained using tabus [52]. To mitigate this problem, it is common to include some diversification technique, such as multiple restarts. When the starting points are provided by and evolved in a genetic algorithm, we obtain a *memetic algorithm* (MA) [16].

Genetic algorithms (GAs) maintain a population of solutions that evolve along generations by applying selection, reproduction and replacement operators. The use of a population of solutions improves the chance of avoiding local optima at the cost of being less capable of intensifying the search in promising areas. In this way, they are complementary to tabu search and this is why hybridising both methods can improve the equilibrium between intensification and diversification by obtaining a synergetic behaviour.

MAs combining different metaheuristics to exploit their complementary search abilities have proved to be very powerful in solving scheduling problems [18, 26, 53]. Thus, to analyse the new objective function and the proposed neigbourhood together with the tabu search procedure, we shall use the

```
Input  A $FJSP$ instance and a feasible solution $s$
Output  A schedule
  $best \leftarrow s$
  $tabuList \leftarrow \emptyset$
  $badIter \leftarrow 0$
  while $badIter < maxIter$ and not in a cycle do
    $\mathcal{N} = \mathcal{N}_{ESD}(s)$
    Sort $\mathcal{N}$ in descending order using $UB$
    $iterESD \leftarrow -\infty$
    for each $s' \in \mathcal{N}$ do
      if $UB(s') > iterESD$ then
        $v \leftarrow$ arc reversed in $s$ to form $s'$
        if $v \notin tabuList$ or $ESD_{avg}(s') > ESD_{avg}(best)$ then
          if $ESD_{avg}(s') > iterESD$ then
            $iterESD \leftarrow ESD_{avg}(s')$
            $s^* \leftarrow s'$
    $v^* \leftarrow$ arc reversed in $s$ to form $s^*$
    Update $tabuList$ with $v^*$
    if $ESD_{avg}(s^*) > ESD_{avg}(best)$ then
      $best \leftarrow s^*$
      $badIter \leftarrow 0$
    else
      $badIter \leftarrow badIter + 1$
    $s \leftarrow s^*$
  return The solution in $best$
```

Algorithm 1: Tabu search algorithm

MA framework proposed in [43] for the fuzzy job shop with due date satisfaction. While keeping the genetic component, we shall replace the objective function (based on aggreement index), the neighbourhood structure and the simple hill climbing method used in that work with ESD_{avg} and the tabu search using the neighbourhood $\mathcal{N}_{ESD}$ proposed here. For the sake of a full and self-contained description of the hybrid solving method, the pseudocode is given in Algorithm 2. We shall refer to the new method as EATS (standing for Evolutionary Algorithm with Tabu Search) in the following.

```
Input  A $FJSP$ instance
Output  A schedule
  Generate random population $P$ with size $popSize$
  Compute $ESD_{avg}$ for each $p_i \in P$
  if $p_L > 0$ then
    Apply Tabu Search to best individual $p_{best} \in P$
    Update genotype of $p_{best}$ (lamarckism)
  for $i = 1, \dots, popSize$ such that $p_i \neq p_{best}$ do
    Apply Tabu Search to $p_i \in P$ with probability $p_L$
    Update genotype of $p_i$ (lamarckism)
  $best \leftarrow$ Best solution $p_{best} \in P$
  $iter \leftarrow 0$
  while $iter < maxIter$ do
    $P' \leftarrow$ Apply selection operator to $P$
    $i \leftarrow 1$
    for $i < popSize$ do
      $(off_1, off_2) \leftarrow$ Apply crossover to $(p'_i, p'_{i+1})$ with probability $p_{cross}$
      if $(p'_i, p'_{i+1})$ not crossed then
        $(off_1, off_2) \leftarrow (p'_i, p'_{i+1})$
      Apply mutation to $off_1$ with probability $p_{mutate}$
      Apply mutation to $off_2$ with probability $p_{mutate}$
      Compute $ESD_{avg}$ for $off_1$ and $off_2$;
      Apply Tabu Search to $off_1$ with probability $p_L$
      Update genotype of $off_1$ (lamarckism)
      Apply Tabu Search to $off_2$ with probability $p_L$
      Update genotype of $off_2$ (lamarckism)
      $(p'_i, p'_{i+1}) \leftarrow$ Apply replacement operator in $(p'_i, p'_{i+1}, off_1, off_2)$
      $i \leftarrow i + 2$
    if $p_L > 0$ then
      Apply Tabu Search to best individual $p'_{best} \in P'$
      Update genotype of $p'_{best}$ (lamarckism)
    if Best solution $p'_{best} \in P'$ is better than $best$ then
      $best \leftarrow p'_{best} \in P'$
      $iter \leftarrow 0$
    else
      $iter \leftarrow iter + 1$
    $P \leftarrow P'$
  return The schedule obtained in $best$
```

Algorithm 2: Evolutionary Algorithm with Tabu Search(EATS)

4. Experimental Results

The purpose of this experimental analysis is threefold. First, to analyse the new objective function ESD_{avg} as a more accurate estimation of the average due-date satisfaction degree than the average agreement index AI_{avg} from the literature. Second, to analyse the behaviour of the proposed EATS hybrid method. Third, to compare it with the state of the art.

For the first purpose we shall measure the quality of ESD_{avg} and AI_{avg} as predictive values under uncertainty of the actual due-date satisfaction that will be obtained after executing the project. To analyse the performance of EATS, after a parametric analysis to find the best configuration of operators and parameters, we will evaluate the proposed neighbour filtering mechanism and the synergy effect between the two components of EATS. Finally, we will compare EATS with other existing methods. On the one hand, we will compare its performance to that of a commercial MILP solver in terms of ESD_{avg} values. On the other hand, direct comparisons with other metaheuristic methods from the literature are not possible, since the objective function to measure due-date satisfaction is new to this work. Instead, we propose indirect comparisons with a method optimising AI as a measure of due-date satisfaction in terms of the actual due-date satisfaction degree obtained by the solutions provided by both algorithms under several scenarios of possible realisations of task durations. To this end, we will consider the MA from [43] which, to the best to our knowledge, constitutes the state-of-the-art for AI_{avg} maximisation.

Throughout this experimental study, we will use the set of the hardest instances considered in [43]. Specifically, fuzzy versions of well-known problems: FT10 (size 10×10), FT20 (20×5), La21, La24, La25 (15×10), La27, La29 (20×10), La38, La40 (15×15), and ABZ7, ABZ8, ABZ9 (20×15). Experiments have been run on a PC with Intel Xeon Gold 6132 processor at 2.6 Ghz and 128 Gb RAM with Linux (CentOS v6.9), using a C++ implementation.

4.1. ESD to measure due-date satisfaction

Under uncertainty, the actual due-date satisfaction may only be known after executing the proposed schedule, when tasks have real durations, that is, on a real scenario out of all the possible ones according to the fuzzy available knowledge. Both *AI* and *ESD* try to predict the satisfaction degree of due dates under all possible scenarios. In order to measure which metric provides a better estimate, we conduct a preliminary set of experiments inspired by the *surrogate δ-robustness* proposed in [43] to measure the error of the estimate made by a predictive schedule built from a random task processing order.

More precisely, let σ be the task processing order associated to a fuzzy solution and let us suppose that tasks are executed in this order so each task starts being processed at the earliest possible instant. After execution, we know the exact processing time p_θ of each task θ, and $\mathbf{p} = \{p_\theta : \theta \in \Theta\}$ constitutes a particular scenario of task durations. The executed schedule corresponds to the crisp schedule we would obtain from σ using a semi-active schedule builder on the crisp problem instance where task durations are given by $\mathbf{p}$. Clearly, p_θ must be in the support of the fuzzy duration $\widehat{p}_\theta$ and, more interestingly, the resulting starting and completion times $s_\theta(\sigma,\mathbf{p})$ and $c_\theta(\sigma,\mathbf{p})$ under scenario $\mathbf{p}$ are in the support of their counterparts of the fuzzy schedule $\widehat{s}_\theta(\sigma)$ and $\widehat{c}_\theta(\sigma)$ for every task $\theta \in \Theta$. For this scenario, it is possible to measure the actual due date satisfaction for the j-th job:

$$sat_j(\sigma,\mathbf{p}) = \mu_{\widetilde{d}_j}(c_j(\sigma,\mathbf{p})), \tag{22}$$

and the average due date satisfaction:

$$DD_{sat}(\sigma,\mathbf{p}) = \frac{1}{n} \sum_{j=1,\dots,n} sat_j(\sigma,\mathbf{p}). \tag{23}$$

Now, the estimation errors made by AI_{avg} and ESD_{avg} for task processing order σ under scenario $\mathbf{p}$ are given by:

$$\delta_{AI}(\sigma,\mathbf{p}) = |AI_{avg}(\sigma) - DD_{avg}(\sigma,\mathbf{p})|, \tag{24}$$

$$\delta_{ESD}(\sigma,\mathbf{p}) = |ESD_{avg}(\sigma) - DD_{avg}(\sigma,\mathbf{p})|. \tag{25}$$

If instead of a single scenario of possible task realisations we have K scenarios, it is still possible to measure the error made both by AI_{avg} and ESD_{avg} under the k-th scenario $\mathbf{p}_k$, denoted $\delta_{AI}(\sigma,\mathbf{p}_k)$ and $\delta_{ESD}(\sigma,\mathbf{p}_k)$, so the average estimation error across all K scenarios will be given by:

$$\delta_{AI}(\sigma) = \frac{1}{K}\sum_{k=1}^{K} \delta_{AI}(\sigma,\mathbf{p}_k), \tag{26}$$

$$\delta_{ESD}(\sigma) = \frac{1}{K}\sum_{k=1}^{K} \delta_{ESD}(\sigma,\mathbf{p}_k). \tag{27}$$

For this experimental analysis, we propose to generate a sample of $K = 1000$ crisp problem instances corresponding to K scenarios of possible realisations of the fuzzy durations. To obtain this sample, for each fuzzy instance we generate $K = 1000$ posible realisations by assigning a deterministic duration to each task which is coherent with the original fuzzy value. Following [54], two samples are generated by Monte-Carlo simulation according to two probability distributions: a uniform probability distribution in the support of the fuzzy duration (Scenario Type I) and the probability distribution obtained for each fuzzy duration after apply the pignistic transformation obtained by considering cuts as uniform distributed probabilities (Scenario Type II).

To have a significant sample of task orderings, for each fuzzy problem instance we generate $L = 30$ feasible processing orders at random σ_l, $1 \le l \le L$ and compute the average estimation errors across K scenarios $\delta_{AI}(\sigma_l)$ and $\delta_{ESD}(\sigma_l)$ for each σ_l. The overall predictive error of *AI* and *ESD* will then be given by the average estimation error across all task orderings and scenarios as follows:

$$\overline{\delta}_{AI} = \frac{1}{L}\sum_{l=1}^{L} \delta_{AI}(\sigma_l), \tag{28}$$

$$\overline{\delta}_{ESD} = \frac{1}{L}\sum_{l=1}^{L} \delta_{ESD}(\sigma_l). \tag{29}$$

These values provide an empirical assessment of the predictive error of *AI* y *ESD* as surrogate values of the average due-date satisfaction after execution. The charts in Figure 8 show the predictive error $\overline{\delta}_{AI}$ and $\overline{\delta}_{ESD}$ for every problem instance under scenarios of Type I and Type II. They clearly show that the new objective function *ESD* proposed in this work provides a more accurate estimate of the average due-date satisfaction of the executed schedule than the classical measure *AI* in both scenario samples for all instances.

4.2. Analysis of EATS performance

4.2.1. Parametric analysis

The experimental study to evaluate the performance of EATS starts with a parametric analysis in order to find the best operator and parameter configuration. We combine an experimental design approach using the Taguchi method with a posterior sequential optimisation strategy [55]. First, the Taguchi method provides a base setup together with parameter order according to their relevance for the algorithm's performance. Then, a refinement of the base setup is obtained following a sequential strategy, tuning one parameter at a time following the order given by the Taguchi method. In this phase we have considered only factors that do not influence the running time, namely, three order-based crossover and mutation operators, together with typical values for crossover and mutation probabilities, and two replacements strategies:

- Crossover operator: Job-Order Crossover (JOX) [56], Generalised Order Crossover (GOX) [57] and Generalised Partially-Mapped Crossover (GPMX) [58]
- Crossover probability: 0.8, 0.9, 1.0 .

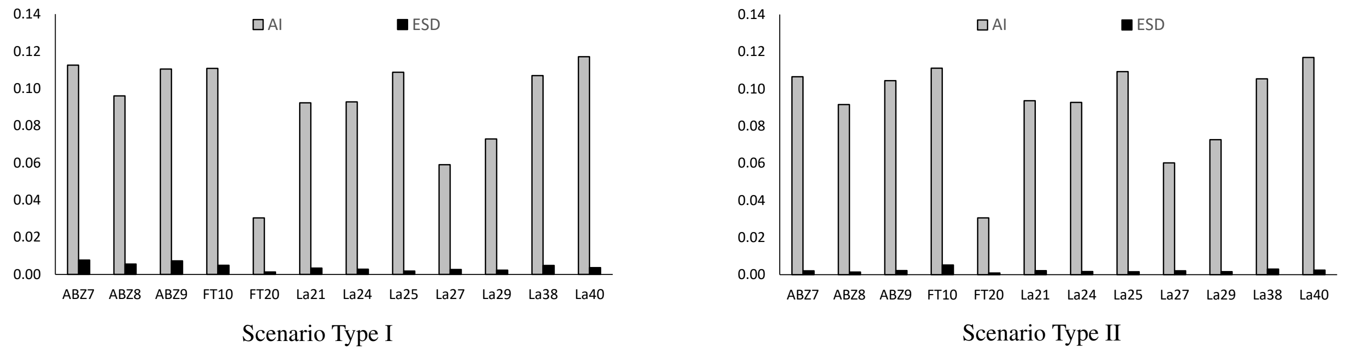


Figure 8: Average values of the estimation errors $\overline{\delta}_{AI}$ and $\overline{\delta}_{ESD}$.

Table 1: Delta values and setup obtained after the parametric study

Parameter	Delta value	Base setup	Final setup
Replacement	0.054	NR	NR
Crossover	1.186	JOX	JOX
Crossover Prob.	0.179	1.00	1.00
Mutation	0.027	IM	SM
Mutation Prob.	0.131	0.15	0.15

- Mutation operator: Insert Mutation (IM), Swap Mutation (SM), and Simple Inversion Mutation (SIM) [59]
- Mutation probability: 0.05, 0.10, 0.15
- Replacement strategy: Allowing repeated individuals (AR), or not (NR)

In all cases, population size is set to 100, the stopping criteria are $maxIter = 25$ and $maxIter = 10$ iterations without improvement for GA and for TS respectively and the probability p_L of applying local search to an individual is 1.

Having four parameters with three levels and one parameter with two levels, the Taguchi method uses the L18 orthogonal array . The resulting "Delta values" shown in Table 1 indicate that the most relevant parameter is the crossover operator, followed by the crossover probability, the mutation probability and, finally, the replacement strategy and mutation operator. Table 1 also shows the "base setup" configuration that results from the Taguchi method. A further sequential tuning phase is undertaken starting from this base configuration and following the order of relevance. The final configuration is reported in Table 1 too. We can see that only the choice of mutation operator changes in this second phase.

There remain to consider two more factors that not only affect the algorithm's performance, but also its runtime: the population size and the level of intensification obtained with the tabu search component.

Figure 9 depicts the evolution of average ESD_{avg} value across 30 runs of EATS along 1000 seconds of runtime when population size is set to 100, 250 and 500 for instance La27, the behaviour on the remaining instances being quite similar. We can see that 1000 seconds exceeds by far the time needed by

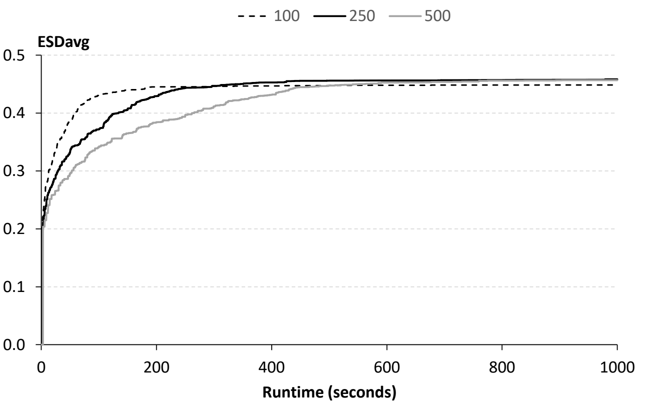


Figure 9: Evolution of EATS with different population sizes on instance La27

EATS to converge in all cases. As expected, the smaller the population size, the faster the algorithm converges. The reason is that in smaller populations there is less diversity to compensate the intensification of the local search. A greater diversity with either 250 or 500 individuals seems to help improve the algorithm's performance: they both converge after approximately 500 seconds to the same ESD_{avg} values, lightly better than those obtained with 100 individuals. In consequence, the population size will be set to 250.

Regarding the intensification pressure provided by the TS component, we have run two experiments to fix the stopping criterion and the proportion of individuals in the population that are improved with TS. First, we have tested different values for the number of iterations that TS is left to run without improvement: 10, 25 and 50. With 25 iterations the average ESD_{avg} value improves 0.3% w.r.t. 10 iterations, but at the cost of consuming over 87% more computational resources; the improvement in ESD_{avg} using 50 iterations instead of 25 is less than 0.2% using over 70% more CPU time. In consequence, we take 10 iterations without improvement as stopping criterion for the TS. Notice however, that in environments where time is not relevant, number of iterations might be increased.

With respect to the probability p_L of applying TS to an individual in the population, we have considered all values in $\{0, 0.25, 0.5, 0.75, 1\}$. Figure 10 shows the graph of convergence for instance La27 along 1000 seconds (more than the

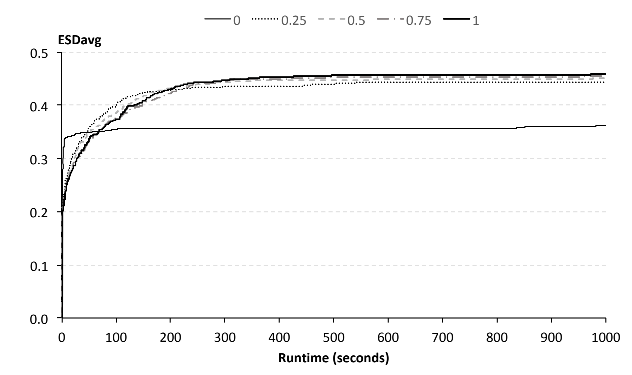


Figure 10: Evolution of EATS with different intensification probabilities on instance La27

Instance	No filter		With filter	
	ESD_{avg}	Time (s)	ESD_{avg}	Time (s)
ABZ7	0.573 (0.007)	900.5	0.577 (0.007)	389.1
ABZ8	0.609 (0.012)	1001.8	0.612 (0.013)	456.5
ABZ9	0.608 (0.016)	1193.6	0.613 (0.022)	541.5
FT10	0.651 (0.000)	32.2	0.651 (0.000)	18.9
FT20	0.269 (0.009)	69.5	0.270 (0.007)	48.5
La21	0.568 (0.011)	188.8	0.571 (0.009)	101.8
La24	0.589 (0.017)	211.0	0.595 (0.017)	106.0
La25	0.582 (0.008)	184.4	0.582 (0.006)	88.5
La27	0.430 (0.030)	472.8	0.448 (0.023)	285.6
La29	0.477 (0.008)	372.2	0.478 (0.008)	194.5
La38	0.774 (0.012)	419.7	0.774 (0.011)	194.3
La40	0.780 (0.001)	298.4	0.780 (0.001)	140.9

Table 2: ESD_{avg} values and runtime (in seconds) obtained by EATS with and without the neighbour filtering mechanism.

time needed to converge, as we have seen above), which is representative of the algorithm's behaviour on all instances. Clearly, the main difference in convergence lies in having some local search ($p_L > 0$) or not having it ($p_L = 0$). We can see that the evolutionary algorithm with no local search experiments a rapid evolution and in the beginning obtains better solutions than when TS is used. The cause is that an important part of runtime is devoted to intensifying the search, particularly in the early generations. However, after 50 seconds, EATS outperforms the evolutionary algorithm with a significant difference in the value of ESD_{avg}. In all cases where TS is applied to some individuals we appreciate a similar convergence pattern, with slightly better performance for $p_L = 1$, i.e., when TS is applied to all individuals in the populations.

In summary, we complement the operator and parameter setup from Table 1 with population size 250, 10 iterations without improvement as stopping criterion for the tabu search and a probability of applying TS to an individual equal to 1.

4.2.2. Synergy and estimators

Having fixed the parameter setting, another series of experiments is conducted in order to evaluate the contribution of the neighbour filtering mechanism based on neighbour's estimation procedure, the tabu search (TS) and evolutionary algorithm (EA) to the performance of the resulting hybrid metaheuristic method EATS.

Table 2 contains average and standard deviation values for ESD_{avg} and CPU time (in seconds) obtained across 30 runs of the EATS on each instance with and without using the ESD estimation algorithm proposed in Section 3.2 to filter non-improving neighbours and avoid unnecessary evaluations as explained in Section 3.3. Clearly, using the upper bound UB of ESD_{avg} to filter neighbours' evaluations translates in a reduction of running time: EATS takes more than twice (108%) on average to run when the filtering mechanism is not used, with runtime increasing up to 131% for ABZ7. However, both approaches obtain solutions of similar quality regarding objective function: results obtained with the filtering mechanism seem slightly better but there is no statistically significant difference except on instances ABZ7 and La27, here in favour of using the estimation-based filter. This shows the great potential of the ESD_{avg} estimate and the associated neighbour filtering mechanism, since it allows the algorithm to have the same or better performance while considerably shortening the runtime (sometimes reducing it by half).

Instance	EA	mTS	EATS
ABZ7	0.530 (0.014)	0.358 (0.015)	**0.577** (0.007)
ABZ8	0.536 (0.023)	0.380 (0.019)	**0.612** (0.013)
ABZ9	0.532 (0.028)	0.362 (0.014)	**0.613** (0.022)
FT10	0.635 (0.008)	0.541 (0.027)	**0.651** (0.000)
FT20	0.243 (0.008)	0.191 (0.008)	**0.270** (0.007)
La21	0.489 (0.014)	0.436 (0.012)	**0.571** (0.009)
La24	0.548 (0.027)	0.444 (0.013)	**0.595** (0.017)
La25	0.539 (0.018)	0.436 (0.012)	**0.582** (0.006)
La27	0.354 (0.020)	0.257 (0.010)	**0.448** (0.023)
La29	0.417 (0.031)	0.244 (0.009)	**0.478** (0.008)
La38	0.699 (0.007)	0.588 (0.017)	**0.774** (0.011)
La40	0.683 (0.022)	0.601 (0.016)	**0.780** (0.001)

Table 3: ESD_{avg} values obtained by EATS and its the main components: EA and LS

To evaluate the synergy effect between the evolutionary algorithm (EA) and the tabu search (TS), EA is run independently for as long as EATS takes to converge, and TS is run as a multi-start tabu search (mTS) with as many restarts as the average number of individuals evaluated by EATS on each instance. The results in Table 3 correspond to the average ESD_{avg} values (and their standard deviation in brackets) obtained on each instance across 30 runs of the three metaheuristic search methods. We can see that EA never surpasses EATS, being near 11% worse in average. A statistical Friedman test to rank means and t-test or Mann-Whitney for every pair of samples (depending on whether or not normality holds) confirm significant differences between both methods. A similar behaviour is encountered with mTS. In the absence of good starting points provided by the EA, results deteriorate (more than 30%) even with longer runtimes (24% longer). A nice synergy effect can be appreciated when both strategies are combined, with EATS obtaining much better results than EA and mTS under equivalent conditions. That is, EATS does benefit from the exploration of EA and also from the intensification of TS.

4.3. Comparison with CPLEX

To assess the difficulty of the problem under consideration and, in particular, of the instances considered herein, the MILP formulation of the problem is used to try to solve the instances using IBM ILOG CPLEX with a time limit of 24 hours (that is, 86400 seconds). In order to demonstrate how both EATS and MILP scale, an additional set of smaller instances are used in this study: the instances proposed in [37](S1-S3 and S5-S7), the instances from [37](S4 and S8) and the instances used in [39](FT06 and La11-14). Three different sizes are overall considered: small instances with 36 tasks, medium instances with 100 tasks and large instances with 150 tasks or more. The results of both EATS and CPLEX are given in Table 4 for the additional set of small instances and in Table 5 for the instances previously considered. For the cases where CPLEX finds a feasible solution but does not prove optimality within the time limit, the gap is given in brackets. A dash means that CPLEX could not find a feasible solution within 24 hours which is the case for 9 instances out of 12 in Table 5. Regarding the remaining instances, the gap value for FT20 is not reported by CPLEX whereas gaps for FT10 and LA24 are 87.20% and 650.00%, respectively.

To take full advantage of the ILOG Concert Technology, it might be interesting to model the piecewise linear functions ESD_j by giving the breakpoints as input, which is then automatically converted by the library into a mixed integer representation. In order to validate this approach, we modify the MILP model (P1) from Section 2.5 so the objective function (8) is replaced with (30) and new binary variables w_j and continuous variables y_j and z_j are added, together with the corresponding constraints. Hence, another MILP model is obtained as (P2). Here, constraints (31)-(32) are the linear formulation of $y_j = \max\{E[\widehat{c}_j], d_j^1\}$ and similarly constraints (33)-(35) represent the linear formulation of $z_j = \min\{y_j, d_j^2\}$. The objective function describes ESD_{avg} with respect to new variables.

(P2)

$$\max \frac{1}{n} \sum_j \frac{d_j^2 - z_j}{d_j^2 - d_j^1} \tag{30}$$

s.t.

$$(9)-(15)$$

$$y_j \geq E[\widehat{c}_j] \quad \forall j \in J \tag{31}$$

$$y_j \geq d_j^1 \quad \forall j \in J \tag{32}$$

$$z_j \geq y_j - W \cdot w_j \quad \forall j \in J \tag{33}$$

$$z_j \geq d_j^2 - W \cdot (1 - w_j) \quad \forall j \in J \tag{34}$$

$$w_j \in \{0,1\} \quad \forall j \in J \tag{35}$$

Tests that are performed with (P1) and (P2) revealed very similar results for large instances, however it is observed that (P2) is faster than (P1) for small instances. For the sake of a fair comparison, the results of (P2) are reported in this section.

The tests reveal that CPLEX finds the optimum for small instances in less than 35 seconds except for S4 which takes 410 seconds, whereas EATS takes 0.86 seconds on average to solve the same instances. CPLEX is able to find feasible solutions of varying quality for medium size instances in 24 hours, but not the optimum. It can find a feasible solution only for one of the large size instances. On the other hand, EATS finds optimal solutions for medium size instances in 7.6 seconds on average. Moreover, it reaches better solutions than CPLEX in less than 250 seconds on average for large size instances.

These results illustrate the complexity of the FJSP with due date satisfaction, which calls for metaheuristic approaches (such as EATS) that might sacrifice optimality guarantees for the sake of finding high quality solutions within reasonable time and computational effort.

Instance	$n \times m$	EATS	MILP
FT06	6x6	**1.000** (0.000)	**1.000** (0.00%)
S1	6x6	**1.000** (0.000)	**1.000** (0.00%)
S2	6x6	**1.000** (0.000)	**1.000** (0.00%)
S3	6x6	**1.000** (0.000)	**1.000** (0.00%)
S4	6x6	**0.866** (0.000)	**0.866** (0.00%)
S5	10x10	**1.000** (0.000)	0.600 (66.67%)
S6	10x10	**1.000** (0.000)	0.900 (11.11%)
S7	10x10	**1.000** (0.000)	0.800 (25.00%)
S8	10x10	**1.000** (0.000)	0.913 (9.49%)
La11	20x5	**1.000** (0.000)	0.900 (11.11%)
La12	20x5	**1.000** (0.000)	0.939 (6.45%)
La13	20x5	**1.000** (0.000)	0.990 (1.01%)
La14	20x5	**1.000** (0.000)	0.941 (6.25%)

Table 4: ESD_{avg} values obtained by EATS and MILP

Instance	$n \times m$	MA	EATS	MILP
ABZ7	20x15	0.563 (0.013)	**0.577** (0.007)	-
ABZ8	20x15	0.590 (0.021)	**0.612** (0.013)	-
ABZ9	20x15	0.577 (0.023)	**0.613** (0.022)	-
FT10	10x10	0.639 (0.009)	**0.651** (0.000)	0.534
FT20	20x5	0.253 (0.011)	**0.270** (0.007)	0.050
La21	15x10	0.556 (0.013)	**0.571** (0.009)	-
La24	15x10	0.571 (0.026)	**0.595** (0.017)	0.133
La25	15x10	0.577 (0.009)	**0.582** (0.006)	-
La27	20x10	0.411 (0.020)	**0.448** (0.023)	-
La29	20x10	0.451 (0.021)	**0.478** (0.008)	-
La38	15x15	0.742 (0.025)	**0.774** (0.011)	-
La40	15x15	0.771 (0.016)	**0.780** (0.001)	-

Table 5: ESD_{avg} values obtained by MA, EATS and MILP

4.4. *Comparison with the state of the art*

To our knowledge, the recent memetic algorithm (MA) from [43] constitutes the state-of-the-art method for average flexible due-date satisfaction in the most challenging fuzzy job shop instances, outperforming prior methods from the literature mentioned in Section 1.1. However, MA and EATS optimise different objective functions (AI_{avg} and ESD_{avg} respectively) even if both try to measure average due-date satisfaction. In consequence, it is not possible to compare their performance directly in terms of objective function values. Instead, we propose to compare the solutions provided by each method with respect to their quality as predictive schedules, measuring the real due-date satisfaction they provide after execution, similarly to what we have done in Section 4.1. We consider, for each instance, the processing orders given by 30 solutions obtained from 30 runs of MA and 30 solutions from EATS. For every fuzzy instance we generate 1000 possible realisations assigning a deterministic processing time to each task according to Scenario Type I and the Scenario Type II and we compute the average due-date satisfaction for each solution across all possible realisations. Figure 11 shows for every instance the boxplots corresponding to the average due-date satisfaction obtained by the 30 solutions from MA and the 30 solutions from EATS when possible realisations correspond to Scenario Type II (the behaviour is similar for Scenario Type I). These plots suggest that EATS provides predictive schedules which perform better in terms of average due-date satisfaction across different possible scenarios. To assess this difference, a t-test is run when samples pass the Kolmogorov-Smirnov test (i.e. they follow a normal distribution) and a Mann-Whitney U test otherwise. The tests show that there is a significant difference in favor of EATS with a significance level of 0.05 for all instances, except FT10. Finally, Table 6 reports for each instance, method and possible scenario type the average values across all 30 solutions and 1000 possible scenarios of the average due-date satisfaction; highlighted in bold are values which are significantly better than their counterpart according to these statistical tests. We may conclude that solutions provided by EATS outperform those provided by MA in all instances except FT10, where both methods behave equally.

In a complementary analysis, we have also implemented MA adapted to optimise ESD_{avg}. However, MA includes a local search component with a neighbourhood structure that is tailored to optimise AI_{avg} and therefore a straightforward run of the algorithm would be unfair for MA. Thus, in this experiment we have replaced its neighbourhood structure by the one introduced in this paper, which is specifically designed for ESD_{avg} maximisation. Since the preliminary runs show that MA is faster than EATS, for the sake of fairness, we have run MA with the original setup from [43] for as long as EATS takes to converge. Table 5 reports the average ESD_{avg} values obtained after 30 runs of each algorithm on each instance (standard deviations are included between brackets). We can see that under equal runtime conditions, EATS outperforms MA in all instances. A Mann-Whitney-Wilcoxon statistical test also confirms that the differences are significant with a significance level of 0.05 in all instances except La25.

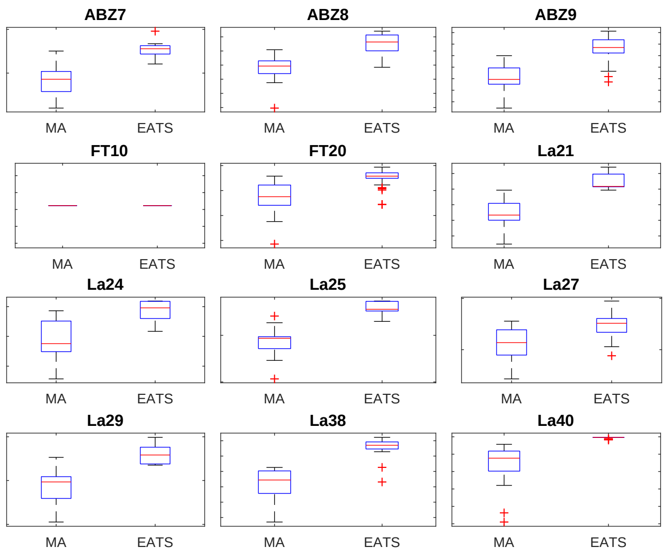


Figure 11: Comparison of solutions obtained with MA and EATS in terms of due-date satisfaction after execution in Scenario Type II

All instances used in this work and the obtained results are available online for future reference[1].

5. Conclusions

We have tackled the job shop scheduling problem with uncertain durations and flexible due dates modelled as fuzzy numbers. We have proposed a new measure of due-date satisfaction and we have defined a new neighbourhood structure purpose-built for this new objective. We have proved results that show that the neighbourhood presents a good theoretical behaviour and we have provided a neighbour estimation mechanism. An evolutionary tabu search method (EATS) has been proposed that uses the neighbourhood structure and a filtering mechanism based on neighbour estimates. Experimental results have been presented that avail the proposal of the new objective function for due-date satisfaction under uncertainty as well as the good behaviour of the metaheuristic search method EATS. Not only have we assessed the synergy effect between the genetic algorithm and the tabu search that compose EATS and the efficiency gain provided by the filtering mechanism, but we have also seen how EATS compares favourably with other approaches from the literature.

Acknowledgements

This research has been supported by the Spanish Government under research grant TIN2016-79190-R and by the Principality of Asturias Government under grant IDI/2018/000176.

[1]Repository section at http://di002.edv.uniovi.es/iscop

Instance	Scenario Type I		Scenario Type II	
	MA	EATS	MA	EATS
ABZ7	0.535 (0.016)	**0.572** (0.008)	0.541 (0.016)	**0.576** (0.008)
ABZ8	0.572 (0.016)	**0.608** (0.013)	0.576 (0.015)	**0.611** (0.013)
ABZ9	0.556 (0.020)	**0.608** (0.021)	0.563 (0.020)	**0.612** (0.022)
FT10	0.646 (0.000)	0.646 (0.000)	0.646 (0.000)	0.646 (0.000)
FT20	0.255 (0.012)	**0.270** (0.007)	0.254 (0.013)	**0.270** (0.007)
La21	0.527 (0.018)	**0.572** (0.009)	0.527 (0.019)	**0.570** (0.009)
La24	0.543 (0.029)	**0.594** (0.017)	0.543 (0.030)	**0.593** (0.017)
La25	0.543 (0.013)	**0.578** (0.005)	0.543 (0.013)	**0.580** (0.006)
La27	0.414 (0.029)	**0.448** (0.023)	0.413 (0.030)	**0.448** (0.023)
La29	0.443 (0.018)	**0.477** (0.009)	0.443 (0.019)	**0.478** (0.009)
La38	0.722 (0.019)	**0.772** (0.012)	0.724 (0.018)	**0.773** (0.012)
La40	0.748 (0.021)	**0.778** (0.002)	0.749 (0.021)	**0.779** (0.001)

Table 6: Average due-date satisfaction across different possible scenarios obtained by MA and EATS

References

[1] M. L. Pinedo, Scheduling. Theory, Algorithms, and Systems., fifth ed., Springer, 2016.

[2] M. A. González, C. Vela, I. González-Rodríguez, R. Varela, Lateness minimization with tabu search for job shop scheduling problem with sequence dependent setup times, Journal of Intelligent Manufacturing 24(4) (2013) 741–54.

[3] J. Kuhpfahl, C. Bierwirth, A study on local search neighbourhoods for the job shop scheduling problem with total weighted tardiness objective, Computers & Operations Research 261 (2016) 44–57.

[4] D. Dubois, H. Fargier, P. Fortemps, Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge, European Journal of Operational Research 147 (2003) 231–52.

[5] D. Dubois, H. Fargier, H. Prade, Fuzzy constraints in job-shop scheduling, Journal of Intelligent Manufacturing 6 (1995) 215–34.

[6] W. Wang, D. Wang, W. Ip, JIT production planning approach with fuzzy due date for okp manufacturing systems, International Journal of Production Economics 58 (1999) 209–15. doi:https://doi.org/10.1016/S0925-5273(98)00122-4.

[7] S. Abdullah, M. Abdolrazzagh-Nezhad, Fuzzy job-shop scheduling problems: A review, Information Sciences 278 (2014) 380–407. doi:10.1016/j.ins.2014.03.060.

[8] J. Behnamian, Survey on fuzzy shop scheduling, Fuzzy Optimization and Decision Making 15 (2016) 331–66. doi:10.1007/s10700-015-9225-5.

[9] H. Prade, Using fuzzy set theory in a scheduling problem: a case study, Fuzzy Sets and Systems 2 (1979) 153–65. doi:10.1016/0165-0114(79)90022-8.

[10] W. K. Wong, C. K. Kwong, P. Y. Mok, W. H. Ip, Genetic optimization of JIT operation schedules for fabric-cutting process in apparel manufacture, Journal of Intelligent Manufacturing 17 (2006) 341–54. doi:10.1007/s10845-005-0007-8.

[11] A. Duenas, D. Petrovic, Multi-objective genetic algorithm for single machine scheduling problem under fuzziness, Fuzzy Optimization and Decision Making 7 (2008) 87–104. doi:10.1007/s10700-007-9026-6.

[12] S. Petrovic, S. Fayad, D. Petrovic, E. Burke, G. Kendall, Fuzzy job shop scheduling with lot-sizing, Annals of Operations Research 159 (2008) 275–92.

[13] S. Niroomand, A. Hadi-Vencheh, N. Mirzaei, S. Molla-Alizadeh-Zavardehi, Hybrid greedy algorithms for fuzzy tardiness/earliness minimisation in a special single machine scheduling problem: case study and generalisation, International Journal of Computer Integrated Manufacturing 29 (2016) 870–88. doi:10.1080/0951192X.2015.1130244.

[14] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Information Sciences 237 (2013) 82–117. doi:10.1016/j.ins.2013.02.041.

[15] C. Blum, J. Puchinger, G. R. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: A survey, Applied Soft Computing 11 (2011) 4135–51. doi:10.1016/j.asoc.2011.02.032.

[16] C. Cotta, L. Mathieson, P. Moscato, Memetic algorithms, in: R. Martí, P. Panos, M. G. C. Resende (Eds.), Handbook of Heuristics, Springer International Publishing, 2016, pp. 1–32. doi:10.1007/978-3-319-07153-4_29-1.

[17] P. Moscato, C. Cotta, An accelerated introduction to memetic algorithms, in: M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, Springer, 2019, pp. 275–309. doi:10.1007/978-3-319-91086-4_9.

[18] J. J. Palacios, J. Puente, C. R. Vela, I. González-Rodríguez, Benchmarks for fuzzy job shop problems, Information Sciences 329 (2016) 736–52. doi:10.1016/j.ins.2015.09.042.

[19] M. A. González, C. R. Vela, R. Varela, Scatter search with path relinking for the flexible job shop scheduling problem, European Journal of Operational Research 245 (2015) 35–45. doi:10.1016/j.ejor.2015.02.052.

[20] M. González, C. R. Vela, R. Varela, An efficient memetic algorithm for the flexible job shop with setup times, in: Proceedings of the 23th International Conference on Automated Planning and Scheduling (ICAPS-2013), 2013, pp. 91–9.

[21] J. J. Palacios, M. A. González, C. R. Vela, I. González-Rodríguez, J. Puente, Genetic tabu search for the fuzzy flexible job shop problem, Computers & Operations Research 54 (2015) 74–89. doi:10.1016/j.cor.2014.08.023.

[22] C. Y. Zhang, P. Li, Y. Rao, Z. Guan, A very fast TS/SA algorithm for the job shop scheduling problem, Computers & Operations Research 35 (2008) 282–94.

[23] C. Smutnicki, W. Bożejko, Tabu search and solution space analyses. the job shop case, in: R. Moreno-Díaz, F. Pichler, A. Quesada-Arencibia (Eds.), Computer Aided Systems Theory – EUROCAST 2017, Springer, 2018, pp. 383–91.

[24] I. Essafi, Y. Mati, S. Dauzère-Pérès, A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem, Computers & Operations Research 35 (2008) 2599–616.

[25] K. Bülbül, A hybrid shifting bottleneck-tabu serach heuristic for the job shop total weighted tardiness problem, Computers & Operations Research 38 (2011) 967–783.

[26] M. A. González, I. González-Rodríguez, C. Vela, R. Varela, An efficient hybrid evolutionary algorithm for scheduling with setup times and weighted tardiness minimization, Soft Computing 16 (2012) 2097–113.

[27] C. Bierwirth, J. Kuhpfahl, Extended GRASP for the job shop scheduling problem with total weighted tardiness objective, European Journal of Operational Research 261 (2017) 835–48.

[28] C. S. McCahon, Using PERT as an approximation of fuzzy project-network analysis, IEEE Transactions on Engineering Management 40 (1993) 146–53.

[29] P. Fortemps, Jobshop scheduling with imprecise durations: a fuzzy approach, IEEE Transactions of Fuzzy Systems 7 (1997) 557–69.

[30] M. Sakawa, T. Mori, An efficient genetic algorithm for job-shop scheduling problems with fuzzy processing time and fuzzy duedate, Computers & Industrial Engineering 36 (1999) 325–41.

[31] D. Lei, Solving fuzzy job shop scheduling problems using random key genetic algorithm, International Journal of Advanced Manufacturing Technologies 49 (2010) 253–62.

[32] O. Engin, M. K. Yilmaz, C. Kahraman, M. E. Baysal, A. Sarucan, A scatter search method for fuzzy job shop scheduling problem with availability constraints, in: Proceedings of the World Congress on Engineering 2011 (WCE 2011), volume II, Newswood Limited, London (U.K.), 2011, pp. 1144–8.

[33] S. Wang, Aorigele, G. Liu, S. Gao, A hybrid discrite imperialist competition algorithm for fuzzy job-shop scheduling problems, IEEE Access 4 (2016) 9320–31. doi:10.1109/ACCESS.2016.2645818.

[34] J. J. Palacios, C. R. Vela, I. González-Rodríguez, J. Puente, A memetic algorithm for due-date satisfaction in fuzzy job shop scheduling, in: IWINAC2017 International Work-Conference on the Interplay Between Natural and Artificial Computation. LNCS 10337, Springer, 2017, pp. 135–45.

[35] Z. Xiang, B. Zhenqiang, W. Guijun, P. Quanke, Optimization of fuzzy job-shop scheduling with multi-process routes and its co-evolutionary algorithm, in: Intelligent Computation Technology and Automation (ICICTA), 2011 International Conference on, volume 1, IEEE, 2011, pp. 866–70. doi:10.1109/ICICTA.2011.618.

[36] C. Fayad, S. Petrovic, A fuzzy genetic algorithm for real-world job-shop scheduling, Innovations in Applied Artificial Intelligence, Lecture Notes in Computer Science 3533 (2005) 524–33.

[37] M. Sakawa, 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 (2000) 393–407.

[38] I. González Rodríguez, J. Puente, C. R. Vela, R. Varela, Semantics of schedules for the fuzzy job shop problem, IEEE Transactions on Systems, Man and Cybernetics, Part A 38 (2008) 655–66.

[39] I. González Rodríguez, J. Puente, C. R. Vela, A multiobjective approach to fuzzy job shop problem using genetic algorithms, CAEPIA 2007, Lecture Notes in Artificial Intelligence 4788 (2007) 80–9.

[40] J. J. Palacios, B. Derbel, On maintaining diversity in MOEA/D: Application to a biobjective combinatorial FJSP, in: GECCO '15 Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, ACM, 2015, pp. 719–26. doi:10.1145/2739480.2754774.

[41] D. Lei, Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems, International Journal of Advanced Manufacturing Technology 37 (2008) 157–65.

[42] C. Wang, N. Tian, Z. Ji, Y. Wang, Multi-objective fuzzy flexible job shop scheduling using memetic algorithm, Journal of Statistical Computation and Simulation 87 (2017) 2828–46. doi:10.1080/00949655.2017.1344846.

[43] J. J. Palacios, I. González-Rodríguez, C. R. Vela, J. Puente, Satisfying flexible due dates in fuzzy job shop by means of hybrid evolutionary algorithms, Integrated Computer-Aided Engineering 26 (2019) 65–84. doi:10.3233/ICA-180583.

[44] S. Heilpern, The expected value of a fuzzy number, Fuzzy Sets and Systems 47 (1992) 81–6.

[45] J. Błażewicz, E. Pesch, M. Sterna, The disjunctive graph machine representation of the job shop scheduling problem, European Journal of Operational Research 127 (2000) 317–31. doi:10.1016/S0377-2217(99)00486-5.

[46] I. González Rodríguez, C. R. Vela, J. Puente, 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), AAAI Press, Sidney, 2008, pp. 124–31.

[47] R. Graham, E. Lawler, J. Lenstra, A. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey., Annals of Discrete Mathematics 4 (1979) 287–326.

[48] P. Van Laarhoven, E. Aarts, K. Lenstra, Job shop scheduling by simulated annealing, Operations Research 40 (1992) 113–25.

[49] M. Dell' Amico, M. Trubian, Applying tabu search to the job-shop scheduling problem, Annals of Operational Research 41 (1993) 231–52.

[50] E. D. Taillard, Parallel taboo search techniques for the job shop scheduling problem, ORSA Journal on Computing 6 (1994) 108–17.

[51] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers & Operations Research 13 (1986) 533–49. doi:10.1016/0305-0548(86)90048-1.

[52] M. Gendreau, J.-Y. Potvin, Tabu search, in: M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, 3rd ed., Springer International Publishing, 2019, pp. 37–55. doi:10.1007/978-3-319-91086-4_2.

[53] C. Cotta, A. J. Fernàndez, Memetic algorithms in planning, scheduling, and timetabling, in: K. P. Dahal, K. C. Tan, P. I. Cowling (Eds.), Evolutionary Scheduling, Springer, 2007, pp. 1–30. doi:10.1007/978-3-540-48584-1_1.

[54] J. J. Palacios, I. González-Rodríguez, C. R. Vela, J. Puente, Robust multiobjective optimisation for fuzzy job shop problems, Applied Soft Computing 56 (2017) 604–16. doi:10.1016/j.asoc.2016.07.004.

[55] E.-G. Talbi, Metaheuristics. From Design to Implementation, Wiley, 2009.

[56] I. Ono, M. Yamamura, S. Kobayashi, A genetic algorithm for job-shop scheduling problems using job-based order crossover, in: Evolutionary Computation, 1996., Proceedings of IEEE International Conference on, IEEE, 1996, pp. 547–52.

[57] C. Bierwirth, A generalized permutation approach to jobshop scheduling with genetic algorithms, OR Spectrum 17 (1995) 87–92.

[58] C. Bierwirth, D. C. Mattfeld, H. Kopfer, On permutation representations for scheduling problems, in: PPSN IV: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, Springer-Verlag, London, UK, 1996, pp. 310–8.

[59] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, S. Dizdarevic, Genetic algorithms for the travelling salesman problem: A review of representations and operators, Artificial Intelligence Review 13 (1999) 129–70. doi:10.1023/A:1006529012972.