algorithms

MDPI

Article

Computing Idle Times in Fuzzy Flexible Job Shop Scheduling

Pablo García Gómez [1], Inés González-Rodríguez [1,*] and Camino R. Vela [2]

[1] Department of Mathematics, Statistics and Computing, Universidad de Cantabria, 39005 Santander, Cantabria, Spain; pablo.garciagomez@unican.es
[2] Department of Computer Science, Universidad de Oviedo, 33204 Gijón, Asturias, Spain; crvela@uniovi.es
* Correspondence: gonzalezri@unican.es

Abstract: The flexible job shop scheduling problem is relevant in many different areas. However, the usual deterministic approach sees its usefulness limited, as uncertainty plays a paramount role in real-world processes. Considering processing times in the form of fuzzy numbers is a computationally affordable way to model uncertainty that enhances the applicability of obtained solutions. Unfortunately, fuzzy processing times add an extra layer of complexity to otherwise straightforward operations. For example, in energy-aware environments, measuring the idle times of resources is of the utmost importance, but it goes from a trivial calculation in the deterministic setting to a critical modelling decision in fuzzy scenarios, where different approaches are possible. In this paper, we analyse the drawbacks of the existing translation of the deterministic approach to a fuzzy context and propose two alternative ways of computing the idle times in a schedule. We show that, unlike in the deterministic setting, the different definitions are not equivalent when fuzzy processing times are considered, and results are directly affected, depending on which one is used. We conclude that the new ways of computing idle times under uncertainty provide more reliable values and, hence, better schedules.

Keywords: scheduling; flexible job shop; fuzzy numbers; idle times

Academic Editor: Yun-Chia Liang

Received: 21 January 2025
Revised: 20 February 2025
Accepted: 24 February 2025
Published: 3 March 2025

Citation: García Gómez, P.; González-Rodríguez, I.; Vela, C.R. Computing Idle Times in Fuzzy Flexible Job Shop Scheduling. *Algorithms* **2025**, *18*, 137. https://doi.org/10.3390/a18030137

1. Introduction

Scheduling is concerned with distributing and planning the execution of multiple tasks on a limited set of resources under different constraints, usually with the goal of optimising some objectives [1,2]. This kind of problem has multiple applications not only in industry and logistics but also in areas such as healthcare, education, or cloud computing [3–7].

The problems in the family known as job shop scheduling are especially significant because they can be used to model many real-world situations [8]. In a job shop, tasks or operations are part of jobs, so operations in a job must be executed in a given order. In the flexible variant, operations can be executed with different resources, so a resource-allocation problem arises in addition to the problem of sequencing operations belonging to different jobs within each resource [9,10]. In addition to presenting a challenge due to its complexity, the flexible job shop problem attracts special attention due to its relevance in modern industrial problems [11,12].

The majority of the research devoted to scheduling assumes that all parameter values are deterministic and well known in advance. However, in real life, it is often the case that some factors, e.g., processing times, are subject to variability due to machine or human influences. This source of uncertainty should not be dismissed; instead, the proper management of the non-deterministic elements of the scheduling process is especially important in Industry 5.0 solutions [13]. In particular, uncertainty in operation processing times can

be addressed with stochastic models or, alternatively, using fuzzy numbers. Following the pioneering works [14,15] fuzzy sets and fuzzy numbers have been amply used in scheduling [16–19]. The reviews [20,21] provide many examples where fuzzy numbers have been successfully applied to shop scheduling problems. In the particular case of the fuzzy flexible job shop scheduling problem, some works consider traditional production-related objectives, for instance, weighted tardiness and earliness [22], makespan [23], or makespan and total workload [24], while others focus on energy efficiency [25,26]. We can also find multiobjective proposals that combine the optimisation of energy consumption together with more traditional objectives, such as makespan or deadline satisfaction [27–30]. A survey of solving methods can be found in [31].

Associated with a schedule are the concepts of idle and waiting times. Idle time is the time spent with the resources waiting to process operations, while the waiting time, its counterpart from the job point of view, is the time spent by jobs waiting for the resources to become available. Idle and waiting times can be of interest for several reasons: they can be part of a constraint, they may constitute an objective in themselves, they may be a component of some other performance objective function, or it may be necessary to compute them as part of a solution method.

Idle and waiting times act as hard constraints when no-wait or no-idle schedules are required. In other cases, a strict no-wait or no-idle constraint might be relaxed and, instead, waiting or idle times need to be kept as small as possible for technological reasons. For example, in food industries, an excessive waiting time after an operation might cause the produced goods to be spoilt and unusable for later operations, or in chemical industries, it can be the case that no waiting time can be allowed between warming and manufacturing operations in the same job. Likewise, it may be undesirable or simply not allowed to have idle resources, for instance, when machines require consumable products, such as paints or powders, that become unusable if the machine stays idle for too long (e.g., paint becomes dried or powder oxidises), or in the steel industry, where furnaces must be kept at a high temperature at all times, regardless of whether they are active or idle. More generally, idle times can be relevant indicators of the efficiency of production systems, especially in those where expensive resources are part of the considered production layout since idle times are directly related to resource utilisation. Also, minimising the total core idle time (i.e., the idle time within the system) is strongly aligned with minimising the makespan, a classical objective in scheduling. Beyond manufacturing, in service environments where the resources are human workers (e.g., in healthcare), it may be desirable to minimise idle times not only to reduce costs and gain efficiency but also to maximise workers' satisfaction [32–34].

Idle times have garnered increasing attention with the growth of energy-aware or green scheduling. According to [35], studies on energy-efficient scheduling are classified into four groups based on energy-saving methods. The first is to reduce unnecessary idle time: it is estimated that, in actual manufacturing processes, machine tools stay idle most of the time and consume about 80% of energy while being in this idle state; in consequence, over 30% of the articles reviewed in [36] consider idle energy consumption as part of the objective function to be minimised. The second approach is to directly turn off the resources when the idle time is long enough and the energy savings compensate for the cost of turning on/off. The third approach is to adjust the machine speed when energy consumption depends on it. Finally, the fourth one is to favour off-peak production: given that, in peak time, electricity costs (and associated pollution) are high, scheduling part of the production at a non-peak time can save a significant amount of electricity cost and mitigate pollution.

We find in the literature many recent examples of green scheduling (many flexible job shop problems) where idle times play an important part. In [35], they consider an energy-efficient flexible job shop under time-of-use pricing and scheduled downtime. With prices of electricity varying over discrete time intervals, the key is to take advantage of idle times and shift the inevitable idle time to peak periods and the production to off-peak periods in order to minimise the energy cost. The flexible job shop problem is also addressed in [37,38]. In the former, the objective is to minimise the total energy consumption by adopting an on/off strategy that takes into account idle times. In the latter, energy consumption is composed of processing energy consumption, idle energy consumption, setup energy consumption, and common energy consumption. A multiobjective dynamic version of the flexible job shop scheduling problem minimising both energy and completion time is tackled in [39] through idle machine time arrangement and machine speed level selection. Inspired by real-world problems, in [40], they present an approach to minimising idle energy consumption and apply it to a heat-intensive process employing a steel-hardening furnace, which requires a high power input to reach and maintain the specific operating temperature, so a considerable amount of energy is consumed even during the periods when no material is being processed. Ref. [41] addresses scheduling in virtual network functions, modelled as a job shop problem, with the goal of simultaneously minimising the makespan and the idle energy loss. In [42] we find another multiobjective scheduling problem, here applied to bakery production, taking into account the reduction in oven idle times as a means of reducing energy consumption. The results show that, by penalising the best makespan by a marginal amount, alternative optimal solutions obtain a considerable reduction in oven idle time by up to 61%, resulting in a strategy that allows small and medium-sized bakeries to lower production costs and reduce CO_2 emissions.

Clearly, to enhance its scope of application and come closer to solving real-world problems, fuzzy scheduling also needs to handle and take into account idle times. However, computing idle times between operations in a resource or the total time a resource is idle is far from trivial in the fuzzy setting since it is not possible to directly extend the equations and techniques from the deterministic framework. This may be the reason why few attempts have been made in this direction, with a few exceptions. Idle times between operations are used to calculate the idle energy consumption in a fuzzy job shop in [43], while multiobjective versions of this problem are tackled in [44,45]. In the first case, idle gaps are used to right-shift operations with the goal of reducing the energy consumption of a given schedule, and in the second case, MIP post-processing makes use of these idle times to also improve the energy cost of a schedule.

Below, we will address the fuzzy flexible job shop problem. We will illustrate the difficulties inherent to computing idle times in the fuzzy framework. We will propose two different ways of computing the idle times between operations in a resource, as well as a third way of directly computing the total idle time in a resource. We will argue that, although, in the deterministic setting these three approaches are equivalent and yield the same value for the total core idle time in a schedule, that is not the case under uncertainty. We will finally conduct an experimental study to empirically evaluate the three proposals for computing idle times in fuzzy schedules.

2. The Fuzzy Flexible Job Shop Problem

The flexible job shop scheduling problem consists in scheduling a set $\mathbf{O}$ of operations (also called tasks) in a set $\mathbf{R}$ of m resources (also called machines) subject to a set of constraints. Operations are organised in a set, $\mathbf{J}$, of n jobs, so operations within a job must be sequentially scheduled. Given an operation, $o \in \mathbf{O}$, $\chi(o)$ will denote the job in $\mathbf{J}$ that this operation belongs to, and $\eta(o)$ will denote the relative position of o in the job, $\chi(o)$. Each

operation, $o \in \mathbf{O}$, can only be processed in resources from a given subset, $\mathbf{R}(o) \subseteq \mathbf{R}$, with a processing time, $p(o, r)$, which is dependent on the resource, $r \in \mathbf{R}(o)$, where it is executed. Furthermore, the following statements are assumed:

1. All jobs and resources are available at the first instant of time.
2. Preemption is not allowed; i.e., each operation must be completed without interruption once started.
3. Machines cannot perform more than one operation at a time.
4. Transportation times between resources and setup times are negligible.
5. Operations are assumed to be successfully completed after the first attempt; the scrap rate is zero.
6. The available capacity of resources is known; there are no stoppages, breakdowns, etc.
7. Buffers with infinite capacity are available for all resources.

Under this problem formulation, the idle time is the time spent by a resource waiting for an operation to process it. The total idle time in a resource can usually be decomposed into front, back, and core idle times. The front idle time is the time a resource waits before starting the first operation (while other resources are already working), the back idle time is the time between the instant when a resource completes its last operation and the instant the last resource finishes working, and the core idle time refers to the time a resource is idle between operations, that is, once it has finished processing one operation and is waiting for the next operation to start being processed.

The objective of this paper is to minimise the total core idle time (that is, the sum of core idle times for all resources) in the fuzzy flexible job shop. In the following subsections, we will describe how to work with fuzzy processing times and how they affect idle time calculations. The notation used throughout the paper is summarised in Table 1.

Table 1. Table of symbols.

Symbol	Description
$\mathbf{O}$	set of operations
$\mathbf{R}$	set of resources
m	number of resources
$\mathbf{J}$	set of jobs
n	number of jobs
$\mathbf{R}(o)$	subset of resources in which operation o can be executed
$\chi(o)$	job to which operation o belongs to
$\eta(o)$	position in which operation o has to be executed relative to its job
$p(o, r)$	processing time of operation o in the resource r
$\boldsymbol{\tau}_\phi$	resource assignments in solution ϕ
$\mathbf{s}_\phi$	operation starting times in solution ϕ
$\tau_\phi(o)$	resource assigned to operation o in solution ϕ
$s_\phi(o)$	position of operation o in $\tau_\phi(o)$
$p_\phi(o)$	processing time of operation o in solution ϕ. Equivalent to $p(o, \tau_\phi(o))$
$c_\phi(o)$	completion time of operation o in solution ϕ
$JP(o)$	job predecessor of operation o
$JS(o)$	job successor of operation o
$RP_\phi(o)$	resource predecessor of operation o in solution ϕ
$RS_\phi(o)$	resource successor of operation o in solution ϕ
$\alpha_\phi(r)$	first operation scheduled in resource r in solution ϕ
$\omega_\phi(r)$	last operation scheduled in resource r in solution ϕ
$\widehat{\square}$	all notations with a hat represent that the number is TFN
$\square^i$	refers to the i-th component of a fuzzy number
$\square^N$	refers to a measure computed using the Naive approach
$\square^K$	refers to a measure computed using the schedule Knowledge approach

Table 1. *Cont.*

Symbol	Description
$\square^C$	refers to a measure computed using the Coarse approach
$I_\phi(o)$	idle time of an operation in ϕ
$CIT_\phi(r)$	core idle time of a resource in ϕ
$TCIT_\phi$	total core idle time of ϕ
$\boldsymbol{\rho}$	a crisp scenario
$f_\phi^{\boldsymbol{\rho}}$	value of the objective function f under the crisp scenario $\boldsymbol{\rho}$ in solution ϕ
$\mathbb{E}[\widehat{a}]$	expected value of $\widehat{a}$
$S(\widehat{a})$	spread of $\widehat{a}$
$MVP(\widehat{a})$	modal value position of $\widehat{a}$
$RDEV(\phi, \boldsymbol{\rho})$	relative distance to the expected value of ϕ with respect to the crisp scenario $\boldsymbol{\rho}$
$UU(\phi, \mathbf{S})$	used uncertainty of solution ϕ in the set of possible scenarios $\mathbf{S}$

2.1. Fuzzy Processing Times

It is fairly common in the literature to model uncertain processing times as triangular fuzzy numbers (TFNs), defined by an interval, $[a^1, a^3]$, of possible values (its support) and a modal value, a^2 [16]. A TFN can be represented as a triplet, $\widehat{a} = (a^1, a^2, a^3)$, so its membership function is given by the following:

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

Arithmetic operations on TFNs are defined based on the Extension Principle [46], so for two TFNs, $\widehat{a}$ and $\widehat{b}$, the sum is a TFN defined as follows:

$$\widehat{a} + \widehat{b} = (a^1 + b^1, a^2 + b^2, a^3 + b^3), \tag{2}$$

And their difference is a TFN defined as follows:

$$\widehat{a} - \widehat{b} = (a^1 - b^3, a^2 - b^2, a^3 - b^1). \tag{3}$$

At first, the resulting expression for the subtraction may seem counter-intuitive. However, notice that the smallest possible value that the uncertain variable, a, described by $\widehat{a}$, may take is a^1, while the greatest possible value for the uncertain variable, b, described by $\widehat{b}$, is b^3 and, hence, the smallest possible value for their difference is $a^1 - b^3$. Analogously, the greatest possible value for the difference will be $a^3 - b^1$. This results in support for the difference where the extreme values of the two involved TFNs can be said to be "crossed". Interestingly enough, the same expressions for the sum and the difference are obtained if we reconstruct them via interval computation from the alpha-cuts [16,46]. A consequence of these definitions is that, in the set of TFNs, the difference is not necessarily the inverse of the sum, that is, $\widehat{a} \neq (\widehat{a} + \widehat{b}) - \widehat{b}$. This is because underlying the definition (3) that results from the Extension Principle is the assumption that the uncertain variables a and b associated with $\widehat{a}$ and $\widehat{b}$ are non-interactive; i.e., they can take values in their support independently of each other. Therefore, when the TFNs $\widehat{a}$ and $\widehat{b}$ are linked, the resulting difference may be an "over-uncertain" or "pessimistic" quantity. This may prove to be an issue in certain applications; it is, for instance, the main difficulty in critical-path analysis and backpropagation in activity networks with fuzzy durations [16].

Unlike the cases of addition and subtraction, the set of TFNs is not closed under the extended maximum. For this reason, in fuzzy scheduling, it is usual to approximate the maximum of two TFNs via interpolation as follows:

$$\max(\widehat{a}, \widehat{b}) \approx (\max(a^1, b^1), \max(a^2, b^2), \max(a^3, b^3)). \tag{4}$$

The approximation always preserves the support and the modal value, and, if the result of the extended maximum is a TFN, it coincides with the approximated value (cf. [47]).

Given that the set of TFNs is not endowed with a natural total order, it is common practice to resort to some ranking method to make comparisons. A widely used one is that based on expected values. For a TFN, $\widehat{a}$, its expected value is as follows:

$$\mathbb{E}[\widehat{a}] = \frac{a^1 + 2a^2 + a^3}{4}, \tag{5}$$

So, any two TFNs $\widehat{a}$ and $\widehat{b}$ can be compared as follows:

$$\widehat{a} \leq_E \widehat{b} \text{ if and only if } \mathbb{E}[\widehat{a}] \leq \mathbb{E}[\widehat{b}]. \tag{6}$$

Accordingly, we will write $\widehat{a} <_E \widehat{b}$ when $\mathbb{E}[\widehat{a}] < \mathbb{E}[\widehat{b}]$. The ranking induced via the expected value coincides with several other rankings from the literature; it is also related to classical interval comparisons via imprecise probabilities, and it has been empirically shown to result in more robust solutions when used in fuzzy scheduling [47,48].

In addition, we consider the following precedence relation between any pair of two TFNs, $\widehat{a}$ and $\widehat{b}$:

$$\widehat{a} \preceq \widehat{b} \text{ if } a^i \leq b^i \; \forall i \in \{1, 2, 3\}, \tag{7}$$

with strong precedence, $\widehat{a} \preccurlyeq \widehat{b}$, if and only if $\widehat{a} \preceq \widehat{b} \wedge \exists i \in \{1, 2, 3\}\; a^i < b^i$.

2.2. Fuzzy Schedules

A solution to the problem, commonly called a schedule and denoted as $\phi = (\boldsymbol{\tau}_\phi, \mathbf{s}_\phi)$, consists of both a resource assignment, $\boldsymbol{\tau}_\phi$, and a starting time assignment, $\mathbf{s}_\phi$, for all operations. This solution is feasible if and only if all precedence and capacity constraints hold. More formally, for an operation, $o \in \mathbf{O}$, let $\tau_\phi(o)$ be the resource to which it is assigned in ϕ, and let $\widehat{s_\phi}(o)$ and $\widehat{c_\phi}(o) = \widehat{s_\phi}(o) + \widehat{p}(o, \tau_\phi(o))$ be, respectively, its starting and completion times in the solution ϕ. Then, precedence constraints hold for ϕ if and only if, for any pair of operations, $o, u \in \mathbf{O}$,

$$\widehat{c_\phi}(o) \preceq \widehat{s_\phi}(u) \text{ when } \chi(o) = \chi(u), \eta(o) < \eta(u) \tag{8}$$

and capacity constraints hold if and only if

$$\tau_\phi(o) \in \mathbf{R}(o) \tag{9}$$

$$\widehat{c_\phi}(o) \preceq \widehat{s_\phi}(u) \vee \widehat{c_\phi}(u) \preceq \widehat{s_\phi}(o) \text{ when } \tau_\phi(o) = \tau_\phi(u). \tag{10}$$

Notice that a feasible starting time assignment, $\mathbf{s}_\phi$, induces an ordering for all the operations in the same resource. In the sequel, given a resource, $r \in \mathbf{R}$, let $\alpha_\phi(r)$ denote the first operation to be processed in r according to schedule ϕ, and let $\omega_\phi(r)$ be the last operation to be processed in r. Also, for an operation, $o \in \mathbf{O}$, let $JP(o)$ (resp. $JS(o)$) denote its immediate predecessor (resp. successor) in its job, $RP_\phi(o)$ (resp. $RS_\phi(o)$) its immediate predecessor (resp. successor) in its resource according to the ordering induced via ϕ, and $\widehat{p_\phi}(o) = \widehat{p}(o, \tau_\phi(o))$ the operation's processing time in the resource to which it is assigned.

A feasible schedule is semi-active if no operation, o, can start earlier (in the sense that it can be assigned another starting time, $\widehat{s}'(o)$, such that $\widehat{s}'(o) \preccurlyeq \widehat{s_\phi}(o)$) without changing the order of processing on any of the resources. In consequence, for a semi-active schedule, the starting time of any operation, o, is the earliest starting time of that operation according to the job and resource processing orders:

$$\widehat{s_\phi}(o) = \max\{\widehat{c_\phi}(JP(o)), \widehat{c_\phi}(RP_\phi(o))\} \tag{11}$$

where $\widehat{c_\phi}(JP(o))$ (resp. $\widehat{c_\phi}(RP_\phi(o))$) is taken to be $(0,0,0)$ if $\eta(o) = \min\{\eta(u) : \chi(u) = \chi(o)\}$, that is, o is the first operation in its job (resp. o is the first operation in its resource, $o = \alpha_\phi(\tau_\phi(o))$).

For so-called regular objective functions (those which are non-decreasing functions of the jobs' completion times), optimal schedules are necessarily semi-active. Assuming semi-active schedules reduces the total number of feasible schedules from infinity to a finite (albeit possibly large) number, with each set of resource orderings providing a single semi-active schedule. For this reason, it is common in the scheduling literature to consider only semi-active schedules. Unless otherwise stated, this will also be assumed here.

2.3. Comparing Fuzzy Schedules

The quality of a schedule, ϕ, is usually measured in terms of a performance or cost function, f_ϕ, so the objective is to find a schedule that minimises f. When working with fuzzy durations, the cost of a fuzzy schedule, ϕ, becomes a TFN $\widehat{f_\phi}$, representing the possible values that the cost, f, may take when ϕ is executed.

When the performance of a schedule, ϕ, is given by a TFN $\widehat{f_\phi}$, the expected value, $\mathbb{E}[\widehat{f_\phi}]$, summarises the schedule's performance in a single crisp value (which allows for comparisons between schedules in the absence of a natural order in the set of TFNs). The quality of a fuzzy schedule ϕ can be further assessed using the measures proposed in [26]. These are concerned, first, with the level of uncertainty in the value $\widehat{f_\phi}$ and, second, with the robustness and accuracy of the fuzzy number $\widehat{f_\phi}$ as a predictor of the actual performance of the schedule when it is executed in a real scenario.

2.3.1. Measures Based on Fuzzy Values

Further insight into a schedule's quality may be gained by considering, in addition to the expected value of fuzzy numbers, two additional features of a TFN $\widehat{a}$:

- The spread (S) of a TFN $\widehat{a}$, given by

$$S(\widehat{a}) = a^3 - a^1, \tag{12}$$

  is a measure of the level of uncertainty in the TFN. Although uncertainty is unavoidable, a solution with a smaller spread in $\widehat{f_\phi}$ should be preferred.
- The modal value position (MVP), defined as

$$MVP(\widehat{a}) = \frac{(a^2 - a^1) - (a^3 - a^2)}{a^3 - a^1}, \tag{13}$$

  is a number in $[-1, 1]$, which is negative when the TFN's modal value a^2 is to the left of the support's midpoint, positive when the modal value is to the right of the midpoint, and zero when the TFN is symmetric. It could somehow be seen as a measure of the skewness of the TFN. Values closer to 0 seem preferable in the sense that they are less biased to either optimistic or pessimistic values.

2.3.2. Measures Based on Possible Crisp Realisations

According to the semantics of fuzzy schedules proposed in [49], a fuzzy schedule, ϕ, is an a priori or predictive solution providing a possibility distribution (in the form of TFNs) for the starting times of all operations and, in consequence, for the values of the resulting performance function, f. When operations are actually executed according to the resource assignment and the ordering induced via the schedule, uncertainty is no longer present: for each operation, o, its executed processing time, $p_\phi(o)$, is a crisp value in support of the fuzzy processing time $\widehat{p}_\phi(o)$; i.e., it is a possible crisp realisation of the fuzzy processing time. The vector of executed processing times, $\boldsymbol{\rho}$, where $\rho(o) = p_\phi(o)$, is a vector of possible realisations of the fuzzy processing times, also called a possible scenario. In consequence, the performance $f_\phi^{\boldsymbol{\rho}}$ of the executed schedule is a crisp value in support of the fuzzy performance $\widehat{f}_\phi$. Thus, the executed schedule can be seen as the a posteriori realisation of the fuzzy schedule ϕ.

Under these semantics, the quality of the a priori fuzzy schedule can be measured relative to its a posteriori performance in a possible scenario or, more generally, across a set of different possible scenarios. To this end, let $\mathbf{S}$ be a set of possible scenarios. We can define the following measures for ϕ:

- Relative distance to the expected value (RDEV) in a possible scenario:

$$RDEV(\phi, \boldsymbol{\rho}) = \frac{f_\phi^{\boldsymbol{\rho}} - \mathbb{E}[\widehat{f}_\phi]}{\mathbb{E}[\widehat{f}_\phi]} \tag{14}$$

  If this value is negative, that is, the cost in the executed schedule is smaller than expected, it means that the a priori schedule objective function is pessimistic. On the contrary, if *RDEV* is positive, the a priori schedule could be seen as optimistic. Overall, the smaller the absolute value of *RDEV*, the closer the predicted performance of the solution to the real performance once the schedule is executed.
- Used uncertainty (UU) in a set of possible scenarios:

$$UU(\phi, \mathbf{S}) = \frac{\max_{\boldsymbol{\rho} \in \mathbf{S}} f_\phi^{\boldsymbol{\rho}} - \min_{\boldsymbol{\rho} \in \mathbf{S}} f_\phi^{\boldsymbol{\rho}}}{f_\phi^3 - f_\phi^1} \tag{15}$$

  This is a value in $[0, 1]$ measuring the proportion of the support of the predicted fuzzy objective function $\widehat{f}_\phi$ that is actually covered across the range of possible scenarios in $\mathbf{S}$. When $\mathbf{S}$ is representative enough of all the possible scenarios, a value of *UU* close to 1 means that the range of values considered as possible by the TFN $\widehat{f}_\phi$ is actually obtainable in real situations. On the other hand, small values of *UU* would mean that the fuzzy schedule incorporates "artificial" uncertainty by considering possible values of f that are in fact never achieved, thus being less informative.

3. Idle Times with Fuzzy Durations

In a deterministic setting, given a schedule, ϕ, it is trivial to compute core idle time before an operation, o: it suffices to subtract the completion time of its immediate predecessor in the resource $RP_\phi(o)$ from the starting time of o. Then, the core idle time for a resource r, $\mathrm{CIT}_\phi(r)$, is obtained by adding the core idle time for all the operations assigned to the resource. Equivalently, $\mathrm{CIT}_\phi(r)$ can be obtained as the difference between the time span since the resource starts processing the first operation until the last operation is finished and the total time spent by the resource executing operations. Finally, the total core idle time, TCIT_ϕ, is the sum of core idle times across all resources $r \in \mathbf{R}$.

Our goal will be to find a schedule minimising the total core idle time. However, extending these definitions to the fuzzy context is not trivial. The cause is the assumption of independence underlying the definition of the fuzzy subtraction, together with the fact that it is not the inverse of the sum (thus losing the equivalence between the two ways of defining the core idle time for a resource).

In the sequel, we propose three different ways of computing the core idle time of a resource, $r \in \mathbf{R}$, given a fuzzy schedule, ϕ, $\widehat{\mathrm{CIT}}_\phi(r)$. In the first two cases, we start by proposing a means to compute the core idle time, $\hat{I}_\phi(o)$, of every operation, o, scheduled in the resource, r, and we then obtain $\widehat{\mathrm{CIT}}_\phi(r)$ as the sum of core idle times across all operations executed in r. In the last case, we adopt a less granular approach and define $\widehat{\mathrm{CIT}}_\phi(r)$ directly as the time span between the starting of the first operation in the resource and the completion of the last operation in the same resource, minus the time spent by the resource processing its operations.

3.1. A Naive Approach

The most straightforward approach to computing the time a resource, r, is idle waiting for an operation, o, under a fuzzy schedule, ϕ, denoted as $\hat{I}_\phi(o)$, is to subtract the completion time of the operation's immediate resource predecessor, $RP_\phi(o)$, from the starting time of o. This is a direct translation of how the idle time can be computed in a deterministic setting, and it has been applied several times in the fuzzy scheduling literature [43–45]. Taking into account that negative values for idle times are impossible, this translates into the following definition:

$$\hat{I}^N_\phi(o) = \begin{cases} (0,0,0) & \text{if } o = \alpha_\phi(\tau_\phi(o)), \\ \max\{(0,0,0), \widehat{s}_\phi(o) - \widehat{c}_\phi(RP_\phi(o))\} & \text{otherwise.} \end{cases} \tag{16}$$

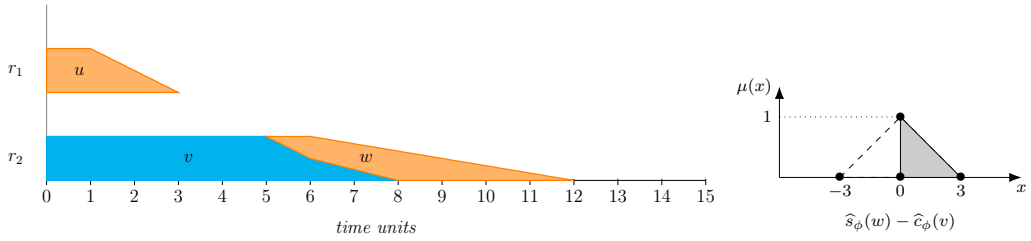
That is, if o is the first operation to be processed in its resource $r = \tau_\phi(o)$, the idle time before o is null, and otherwise, it is the difference between the operation's starting time and its predecessor's completion time, discounting negative values (which we know are impossible). Then, the core idle time for resource r is given by the sum of idle time for the operations scheduled in r:

$$\widehat{\mathrm{CIT}}^N_\phi(r) = \sum_{o \in \mathbf{O}: \tau_\phi(o) = r} \hat{I}^N_\phi(o) \tag{17}$$

The problem with this straightforward extension of the deterministic definition is that, as mentioned in Section 2.1, the fuzzy subtraction assumes that the variables involved are non-interactive. Making such an assumption for two consecutive operations in a resource is at least naive and, in most cases, problematic. Indeed, in a feasible schedule, an operation, o, cannot start before its immediate processor is completed, so there is a clear dependence between $\widehat{s}_\phi(o)$ and $\widehat{c}_\phi(RP_\phi(o))$. This dependence is highly marked in the case of semi-active schedules, where $\widehat{s}_\phi(o)$ is defined in terms of $\widehat{c}_\phi(RP_\phi(o))$, as shown in Equation (11). As a result, computing the idle time for an operation, o, using (16) will most likely result in a quantity with added artificial uncertainty in the sense that many values in its support are actually unattainable. Let us illustrate this with an example:

Let u, v, and w be the only three operations in a schedule ϕ where u, the job predecessor of w, is processed on its own in a resource, while v is the resource predecessor of w in a second resource. Let the processing times of these tasks be $\widehat{p}_\phi(u) = (1,2,3)$, $\widehat{p}_\phi(v) = (5,6,8)$, and $\widehat{p}_\phi(w) = (1,3,4)$. Clearly, if u and v start at instant 0, their completion times are, respectively, $\widehat{c}(u) = (1,2,3)$ and $\widehat{c}(v) = (5,6,8)$. Hence, the completion time of u cannot affect the starting time of w, $\widehat{s}_\phi(w)$, which, for any feasible schedule, must be such that $\widehat{c}(v) \preceq \widehat{s}(w)$, with an equality for semi-active schedules, i.e., $\widehat{s}(w) = \widehat{c}(v) = (5,6,8)$. The Gantt chart of

the schedule, adapted to TFNs, as proposed in [50], can be seen in Figure 1. Now, if we compute the difference, $\widehat{s}_\phi(w) - \widehat{c}_\phi(v) = (5-8, 6-6, 8-5) = (-3, 0, 3)$. This number is counterintuitive for two reasons. First, allowing for negative idle time values does not make any sense. This is solved by taking the maximum with, $(0, 0, 0)$; so, according to (16), the idle time for w is $I^N(w) = (0, 0, 3)$. We depict the resulting TFN in Figure 1b, where the dashed line represents the negative side that is removed, with the area under the membership function for $I^N(w)$ shaded in grey. The second reason is that common sense dictates that the idle time between two contiguous operations should be zero, as the absence of a gap between v and w in Figure 1 illustrates, but even after eliminating negative values, $I^N(w)$ considers as possible idle times as large as 3. The source of these issues is the wrong underlying assumption that $\widehat{s}(w)$ and $\widehat{c}(v)$ are independent; so, for instance, $s(w)$ could be 5 when $c(v)$ is 8 (even if we know this is impossible in a feasible schedule). That is, the independence assumption introduces artificial uncertainty in the resulting difference.



(**a**) Gantt diagram of the example schedule. (**b**) TFN of $\hat{I}^N_\phi(w)$.

Figure 1. Example to illustrate idle time calculations.

3.2. An Approach Incorporating Schedule Knowledge

If we consider again the previous example, the problem stems from the fact that $\widehat{s}_\phi(w)$ and $\widehat{c}_\phi(v)$ are linked variables. Indeed, the starting time of w is computed according to Equation (11); that is, $\widehat{s}_\phi(w) = \max\{\widehat{c}_\phi(u), \widehat{c}_\phi(v)\}$. Substituting in the difference component of the idle time (16),

$$\widehat{s}_\phi(w) - \widehat{c}_\phi(v) = \max\{\widehat{c}_\phi(u), \widehat{c}_\phi(v)\} - \widehat{c}_\phi(v).$$

Now, given the way in which the difference and the maximum are defined in (3) and (4), we can rearrange the elements in this equation, so

$$\hat{I}^N_\phi(w) = \max\{(0, 0, 0), \widehat{c}_\phi(u) - \widehat{c}_\phi(v), \widehat{c}_\phi(v) - \widehat{c}_\phi(v)\}.$$

Of course, in a fuzzy schedule, $\widehat{c}_\phi(v)$ is an uncertain quantity, but we know that the underlying variable cannot take two different values in the same schedule, so, in this context, $\widehat{c}_\phi(v) - \widehat{c}_\phi(v)$ must be $(0, 0, 0)$, which means that the idle time for w should, in fact, be equal to $\max\{(0, 0, 0), \widehat{c}_\phi(u) - \widehat{c}_\phi(v)\}$.

Notice that this expression makes sense from the scheduling point of view since it means that the idle time is the time difference between the completion time of the job predecessor and the resource predecessor, i.e., the time the resource has to wait once it has finished processing the resource predecessor of operation, o, until the other predecessor of o is completed. This suggests an alternative definition of the idle time of an operation, o, which takes into account our knowledge of schedules:

$$\hat{I}_{\phi}^{K}(o) = \begin{cases} (0,0,0) & \text{if } \eta(o) = \min\{\eta(u) : \chi(u) = \chi(o)\} \\ & \quad \vee o = \alpha_{\phi}(\tau_{\phi}(o)) \\ & \quad \vee \tau_{\phi}(JP(o)) = \tau_{\phi}(RP(o)) \\ \max\{(0,0,0), \hat{c}_{\phi}(JP(o)) - \hat{c}_{\phi}(RP_{\phi}(o))\} & \text{otherwise} \end{cases} \tag{18}$$

That is, in general, we take the idle time for o to be the difference between the completion times of its predecessors in the resource and the job (discounting negative values), with three exceptions. In the case that o is either the first operation in its job or the first operation to be processed in its resource, the idle time will be $(0,0,0)$. Additionally, unlike in the traditional job shop, in the flexible variant, it is possible that both the resource and the job predecessor are executed in the same resource, that is, $\tau_{\phi}(JP(o)) = \tau_{\phi}(RP(o))$; due to resource constraints, one must be executed before the other, and operation o can thus start as soon as the latest predecessor is completed, resulting in no idle time.

Once the idle time for an operation is defined, the core idle time for resource r is again given by the sum of idle time for the operations scheduled in r:

$$\widehat{\mathrm{CIT}}_{\phi}^{K}(r) = \sum_{o \in \mathbf{O}: \tau_{\phi}(o) = r} \hat{I}_{\phi}^{K}(o) \tag{19}$$

Let us go back to the toy example of Section 3.1; if the idle time of operation w is computed using (18), we have $\hat{I}_{\phi}^{K}(w) = \max\{(1-30, 2-25, 3-20), (0,0,0)\} = (0,0,0)$. This results in a TFN without added artificial uncertainty, as expected.

3.3. A Coarser Approach

Both the expressions from (17) and (19) correspond to a fine-grained approach where idle times are computed for each operation, and then they are aggregated to obtain the resource core idle time. However, it is sometimes the case that individual idle times for operations are of no interest, and only the resource idle time as a whole is necessary. In these situations, where not so much granularity is necessary, it is possible to compute the core idle time for resource r in a coarser manner, as follows:

$$\widehat{\mathrm{CIT}}_{\phi}^{C}(r) = \begin{cases} (0,0,0) & \text{if } \{o \in \mathbf{O} : \tau_{\phi}(o) = r\} = \emptyset \\ \max\{(0,0,0), \left(\hat{c}_{\phi}(\omega_{\phi}(r)) - \hat{s}_{\phi}(\alpha_{\phi}(r))\right) - \widehat{W}_{\phi}(r)\} & \text{otherwise.} \end{cases} \tag{20}$$

where

$$\widehat{W}_{\phi}(r) = \sum_{o \in \mathbf{O}: \tau_{\phi}(o) = r} \hat{p}_{\phi}(o) \tag{21}$$

is the workload for resource r, that is, the time in which it is active processing operations, defined as the sum of the processing times of all operations assigned to the resource.

The rationale behind this expression, assuming that at least one operation is scheduled in resource r, is to calculate the time span between the starting time of the first operation to be processed in the resource and the completion time of the last operation in that resource and then subtract the total time in which the resource is actually active. Again, we discount negative values, which we know are not possible, and the result will be the core idle time of the resource.

3.4. The Total Core Idle Time

Regardless of how the core idle time, $\widehat{\mathrm{CIT}}_{\phi}(r)$, for each resource, $r \in \mathbf{R}$, is computed, the total core idle time of a schedule is obtained as the sum across all resources of their core idle time:

$$\widehat{\mathrm{TCIT}}_\phi = \sum_{r \in \mathbf{R}} \widehat{\mathrm{CIT}}_\phi(r). \tag{22}$$

We may add a superscript, N, K, C, if we want to highlight that this function is computed using (17), (19), or (20), respectively.

Let us illustrate the three different ways in which $\widehat{\mathrm{CIT}}$ can be computed with a toy example. Suppose that we have the schedule ϕ depicted in Figure 2, comprising eight operations scheduled in three resources. To represent the different values of the operations in the schedule, we will use grids as the following:

$$\begin{bmatrix} o_2 & o_8 & \\ o_1 & o_7 & o_5 \\ o_4 & o_6 & o_3 \end{bmatrix}$$

where each row corresponds to the operations scheduled in a resource. This way, the processing times of operations are as follows:

$$\begin{bmatrix} (30,30,40) & (10,10,10) & \\ (20,30,40) & (10,20,30) & (35,40,40) \\ (15,20,30) & (20,20,40) & (10,30,30) \end{bmatrix}$$

Their starting times are given by the following:

$$\begin{bmatrix} (20,30,40) & (50,60,100) & \\ (0,0,0) & (35,40,70) & (45,60,100) \\ (0,0,0) & (15,20,30) & (50,60,80) \end{bmatrix}$$

And their completion times are given by:

$$\begin{bmatrix} (50,60,80) & (60,70,110) & \\ (20,30,40) & (45,60,100) & (80,100,140) \\ (15,20,30) & (35,40,70) & (60,90,110) \end{bmatrix}$$

If we compute operations' idle times, $\widehat{I}^N_\phi$, using the naive approach from (16), we obtain the following values:

$$\begin{bmatrix} (0,0,0) & (0,0,50) & \\ (0,0,0) & (0,10,50) & (0,0,55) \\ (0,0,0) & (0,0,15) & (0,20,45) \end{bmatrix}$$

This results in resource core idle times $\widehat{\mathrm{CIT}}^N_\phi(r_1) = (0,0,50)$, $\widehat{\mathrm{CIT}}^N_\phi(r_2) = (0,10,105)$, and $\widehat{\mathrm{CIT}}^N_\phi(r_3) = (0,20,60)$ and, thus, a total core idle time,

$$\widehat{\mathrm{TCIT}}^N_\phi = (0,30,215).$$

If operation idle times are instead computed with the knowledge-based approach, as proposed in (18), we obtain the following values of $\widehat{I}^K_\phi$ for all operations:

$$\begin{bmatrix} (0,0,0) & (0,0,50) & \\ (0,0,0) & (0,10,50) & (0,0,0) \\ (0,0,0) & (0,0,0) & (0,20,45) \end{bmatrix}$$

These yield resource core idle times $\widehat{\text{CIT}}_{\phi}^{K}(r_1) = (0, 0, 50)$, $\widehat{\text{CIT}}_{\phi}^{K}(r_2) = (0, 10, 50)$, and $\widehat{\text{CIT}}_{\phi}^{K}(r_3) = (0, 20, 45)$, so the total core idle time will be

$$\widehat{\text{TCIT}}_{\phi}^{K} = (0, 30, 145).$$

Finally, using the coarse approach to directly compute the core idle time of resources as per (20), we obtain $\widehat{\text{CIT}}_{\phi}^{C}(r_1) = (0, 0, 50)$, $\widehat{\text{CIT}}_{\phi}^{C}(r_2) = (0, 10, 75)$, and $\widehat{\text{CIT}}_{\phi}^{C}(r_3) = (0, 20, 65)$. Then, the total core idle time according to this method is

$$\widehat{\text{TCIT}}_{\phi}^{C} = (0, 30, 190).$$

We can appreciate how, unlike the deterministic case, in the fuzzy framework, each approach yields a different value for $\widehat{\text{TCIT}}$. The three different TFNs can be seen in Figure 3. It is clear that the support (and, hence, the spread) of the resulting TFN is different in each case. Given their definitions, it is guaranteed that the spread of $\widehat{\text{TCIT}}_{\phi}^{K}$ will always be less than or equal to the spread of $\widehat{\text{TCIT}}_{\phi}^{N}$, i.e., computing the total core idle time with the knowledge-based approach accumulates less uncertainty, as illustrated by the example. In this particular case, the spread for $\widehat{\text{TCIT}}_{\phi}^{C}$ is also larger than the one for $\widehat{\text{TCIT}}_{\phi}^{K}$. However, it is not guaranteed that this will always be so, as shown in the experimental results in Section 4. The example also illustrates that the modal value of $\widehat{\text{TCIT}}_{\phi}$ is always the same, regardless of the approach used to compute it. This is a direct consequence of how arithmetic operations are performed on TFNs. Finally, notice how, by truncating negative values, in all three cases, uncertainty accumulates towards the right-hand side, resulting in skewed TFNs.

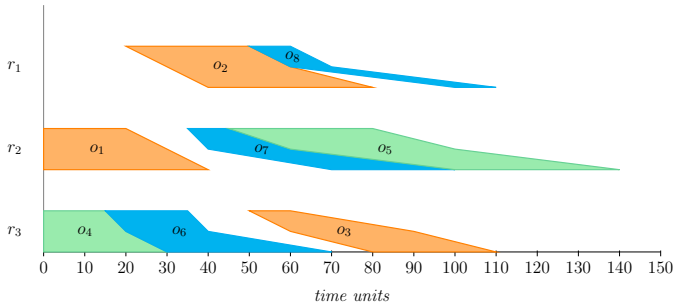
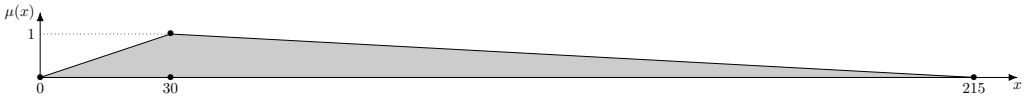
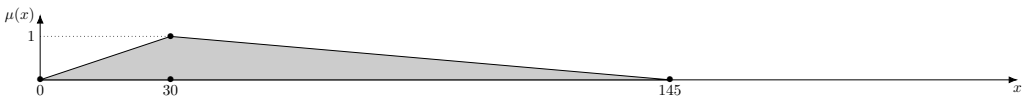


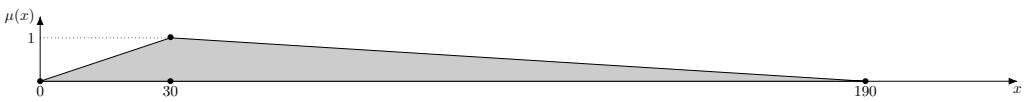
Figure 2. Schedule to illustrate the total core idle time calculation.



(**a**) $\widehat{\text{TCIT}}_{\phi}^{N}$



(**b**) $\widehat{\text{TCIT}}_{\phi}^{K}$



(**c**) $\widehat{\text{TCIT}}_{\phi}^{C}$

Figure 3. $\widehat{\text{TCIT}}_{\phi}$ obtained using the three different approaches for the example in Figure 2.

4. Results

The objective of the experimental study is to empirically assess the three different approaches introduced in Section 3 in order to compute the total core idle time for fuzzy schedules. We will use 12 benchmark instances from the literature [25]: 07a, 08a, 09a, 10a, 11a, and 12a with 15 jobs, 8 resources, and 293 operations (denoted as $15 \times 8 \times 293$ in the following) and 13a, 14a, 15a, 16a, 17a, and 18a of size $20 \times 10 \times 387$. In instances 07a, 08a, 09a, 13a, 14a, and 15a, operations have the same processing time, regardless of the resources where they are executed, while instances 10a, 11a, 12a, 16a, 17a, and 18a have resource-dependent processing times. There are also three increasing flexibility levels, with 07a, 10a, 13a, and 16a being those instances with the lowest flexibility and 09a, 12a, 15a and 18a being those instances with the highest flexibility.

The study follows the methodology proposed in [26]. For each instance, schedules will be generated and compared in terms of their performance regarding $\widehat{\text{TCIT}}$ using first the measures introduced in Section 2.3.1 and then those introduced in Section 2.3.2. In both cases, this will be done with two types of experiments: over random solutions and with solutions obtained with a search algorithm. The goal of the random-solution experiments is to evaluate the three alternatives for computing $\widehat{\text{TCIT}}$ without the possible bias that may be introduced via any solver, whereas the searched-solution experiments pretend to compare the three models when embedded in a search algorithm, with the objective of evaluating their capability of guiding the search towards good schedules. In both sets of experiments, 100 schedules are obtained per instance; notice that, for the searched-solution comparisons, this means generating 100 schedules for each of the three models ($\widehat{\text{TCIT}}^N$, $\widehat{\text{TCIT}}^K$, and $\widehat{\text{TCIT}}^C$) under consideration.

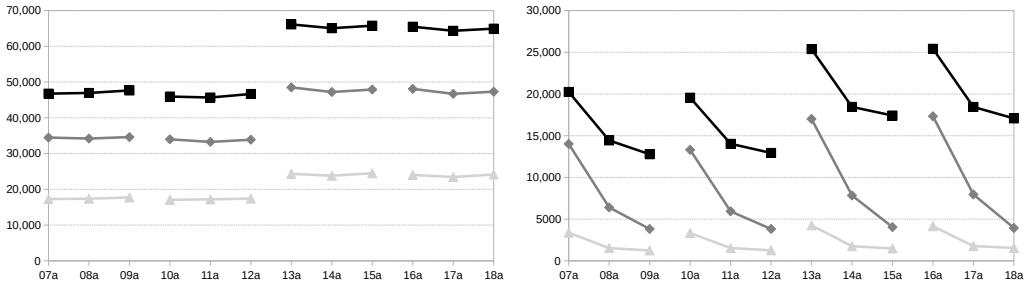
As a solution method for the searched-solution comparisons, we employ the tabu search algorithm proposed in [51] for the classical job shop, extending the neighbourhood function so that every operation is moved to all feasible positions both in its current resource and in others where it can be executed. This is probably the most naive neighbourhood available, as it tries to reschedule every operation in every position, but at the same time, it ensures connectivity and is agnostic to the models, minimising their influence on the results. At every iteration, a tabu search generates all possible neighbours and selects the best one that is not tabu. Then, it stores the inverse of the move performed to obtain the selected neighbour in a memory structure, called the tabu list, so, in the next iterations, that move is forbidden, thus avoiding the undoing of recently made moves. In this way, the tabu search is able to escape local optima. In addition, the tabu list also serves as a diversification mechanism, as different movements are encouraged. A detailed pseudocode can be found in Figure A1. The tabu search parameters are set to the values used in [26] for the same set of instances. This algorithm will be used in all experiments, with the only change being the objective function, which will vary in order to test the different proposals.

4.1. Comparisons Based on Fuzzy TCIT *Values*

As mentioned in Section 2.3.1, fuzzy TCIT values can be compared in terms of their expected value, their spread, S, and their modal value position, MVP.

Figure 4 shows the average $\mathbb{E}[\widehat{\text{TCIT}}]$ values obtained for each instance in the two types of experiments. Figure 4a corresponds to the average values obtained when evaluating the random schedules with the three models for computing $\widehat{\text{TCIT}}$. Figure 4b depicts average values obtained when evaluating the schedules obtained with the tabu search using each of the three models. In both experiments, there is clearly a difference between the three methods for computing the fuzzy TCIT: $\widehat{\text{TCIT}}^C$ yields better average expected values than $\widehat{\text{TCIT}}^K$, and this, in turn, yields better values than $\widehat{\text{TCIT}}^N$.

Furthermore, for the random solutions experiment, the variation in the expected value seems to depend only on the problem size and not on the flexibility level, as illustrated by the flat lines. This makes sense since, the larger the number of operations, the larger the number of possible gaps, and, as they are not optimised in any way, the longer the total idle time. However, for the searched-solution experiment, differences exist depending not only on the instance size but also on the flexibility level. This may be because greater flexibility provides the search algorithm with more opportunities to move operations between resources and fill inactivity gaps, thus reducing the total core idle time. It is remarkable that, here, $\widehat{\text{TCIT}}^{K}$ seems to yield better improvements when greater flexibility is introduced, coming closer to $\widehat{\text{TCIT}}^{C}$. On the other hand, the improvement of the naive approach $\widehat{\text{TCIT}}^{N}$ does not seem to be as pronounced when flexibility increases. Also, $\widehat{\text{TCIT}}^{K}$ expected values are significantly higher than $\widehat{\text{TCIT}}^{C}$. This is expected because, although $\widehat{\text{TCIT}}^{K}$ has been defined to reduce artificial uncertainty due to dependencies between operations, there are still relations that we cannot easily account for. Thus, the sum of idle times between operations, that is the sum of multiple subtractions, is going to have predictably more artificial uncertainty than the coarser approach, which only makes one subtraction, as the results demonstrate.



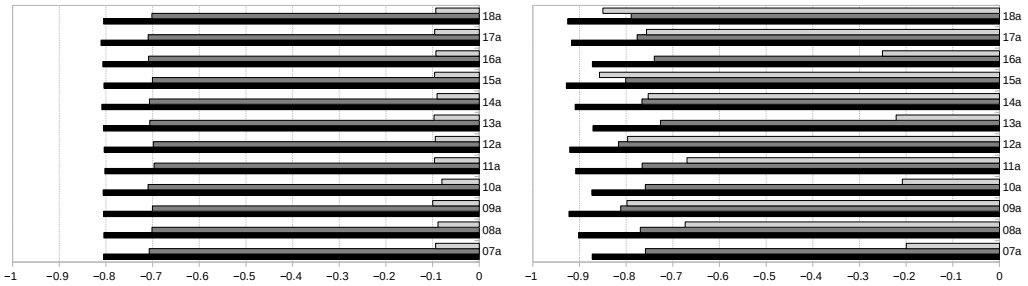
(**a**) Random solutions $\mathbb{E}[\widehat{\text{TCIT}}]$ (**b**) Searched solutions $\mathbb{E}[\widehat{\text{TCIT}}]$

Figure 4. $\mathbb{E}[\widehat{\text{TCIT}}]$ values (y-axis) obtained for all instances (x-axis), with the black line representing $\widehat{\text{TCIT}}^{N}$, the darker grey one $\widehat{\text{TCIT}}^{K}$, and the lighter grey one $\widehat{\text{TCIT}}^{C}$. Instances with the same size and resource dependency in processing times are grouped so that increasing flexibility levels are joined by a line. Please notice that the scale changes between graphs.

When the spread is considered, the obtained graphs are very similar. The reason is that, in these benchmark instances, operations have similar-magnitude processing times with different values for each of the three components of the TFNs. As computing starting times according to Equation (11) are accumulative, the greater the number of already-scheduled operations, the larger the expected value of the TFN and the difference between the first and third components. Then, as the subtraction according to Equation (3) is computed by crossing the value of the lower and upper values of the two TFNs involved, it is predictable that larger expected values also mean larger spread values.

Finally, Figure 5 shows the MVP values obtained in the experiments. Notice that, both in the random solutions and the searched solutions experiments, MVP is negative, meaning that the modal point is situated to the left of the support's midpoint; i.e., the TFNs are skewed to the right. The main reason for this behaviour is that, in all functions, we are removing negative values because negative times are not possible, but this trimming will only take place on the left side of TFNs. Also, notice that, in the searched-solution experiments, the absolute value of MVP is closer to -1 compared to the random solutions. This is because better solutions mean smaller gaps between operations in the same resource, and then, the probability for the predecessor in the resource being the operation defining the starting time is higher, introducing a dependency between these operations that increases

the chances of having negative numbers for idle times. However, the conclusion should not be that being closer to -1 is better. The reason is that the processing times of operations are almost symmetrical, but $\widehat{\text{TCIT}}$ does not maintain this symmetry, which means that we are accumulating artificial uncertainty on one of the sides. Notice that this effect is aggravated when there is higher flexibility, and that it is much more pronounced for $\widehat{\text{TCIT}}^{C}$. This phenomenon can be illustrated as follows. Suppose that, during the tabu search, there is a resource where an operation can be scheduled without altering the starting time of the first operation already scheduled in the resource or the completion time of the last operation in the resource. Then, the minuend in (20) (i.e., the time span between the start of the first operation and the end of the last one) is not altered, while the subtrahend, the sum of processing times of operations assigned to the resource, grows. In consequence, the first component of the difference will be smaller, more likely negative, and, therefore, more likely to be truncated to 0, resulting in a TFN more skewed to the right. When the flexibility level is low, so is the possibility of moving operations to a different resource, while higher flexibility increases the options to move operations between resources, leading to the described behaviour.



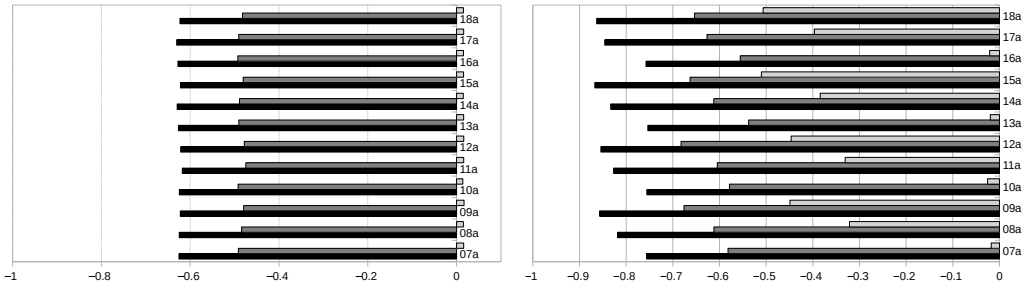
(**a**) Random solutions $MVP(\widehat{\text{TCIT}})$ (**b**) Searched solutions $MVP(\widehat{\text{TCIT}})$

Figure 5. $MVP(\widehat{\text{TCIT}})$ values (x-axis) obtained for all instances (y-axis) using $\widehat{\text{TCIT}}^{N}$ (in black), $\widehat{\text{TCIT}}^{K}$ (in dark grey), and $\widehat{\text{TCIT}}^{C}$ (in light grey).

4.2. Comparisons Based on Crisp Scenarios

We now evaluate the three models for computing $\widehat{\text{TCIT}}$ using the measures described in Section 2.3.2. To this end, we will use a unique set of 1000 crisp possible scenarios, **S**, derived from the fuzzy instances. These scenarios are generated by setting the processing time of every operation in each allowed resource to a random value derived from a uniform distribution in support of the TFNs.

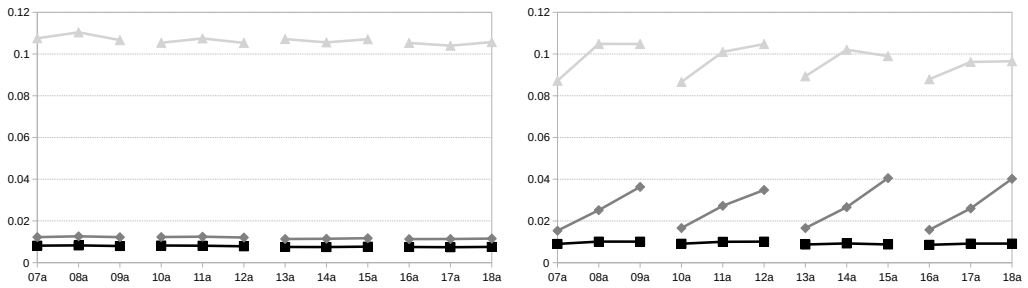
Figure 6 shows, for each instance, the average $RDEV$ across all scenarios in **S**, both for the random and searched solutions experiments. Clearly, both the results obtained with $\widehat{\text{TCIT}}^{K}$ and $\widehat{\text{TCIT}}^{C}$ on the crisp scenarios are closer to the predicted expected value than those obtained with $\widehat{\text{TCIT}}^{N}$. Those obtained via the coarser approach are the closest to the expected value, evidencing the lower amount of artificial uncertainty that it introduces. One remarkable detail, when comparing random- and searched-solution cases for $\widehat{\text{TCIT}}^{C}$, is that, in the random-solution experiment, the relative distance values are positive; i.e., the TCIT values obtained with the executed schedules are, on average, larger than expected. In other words, the predicted estimate provided via $\widehat{\text{TCIT}}^{C}$ is slightly optimistic. However, when integrated into the tabu search, the relative deviation becomes negative. As was the case with MVP, this can be explained because truncating negative values (which are more likely after an improving local search move) cause the $\widehat{\text{TCIT}}$ to be skewed to the right, with a subsequent deviation in the expected values.



(**a**) Random solutions $RDEV$ (**b**) Searched solutions $RDEV$

Figure 6. Average $RDEV(\widehat{\text{TCIT}})$ values (x-axis) obtained for all instances (y-axis) using $\widehat{\text{TCIT}}^{N}$ (in black), $\widehat{\text{TCIT}}^{K}$ (in dark grey) and $\widehat{\text{TCIT}}^{C}$ (in light grey).

Finally, Figure 7 depicts the results obtained when considering the used uncertainty measure, UU. For random solutions, there is no difference in UU with different flexibility levels. However, in the case of searched solutions, there exist interesting differences between the three approaches. In the case of $\widehat{\text{TCIT}}^{K}$, increasing flexibility also causes the used uncertainty to increase. However, for $\widehat{\text{TCIT}}^{N}$, increasing flexibility does not make any difference in either the random or searched solutions. The case of $\widehat{\text{TCIT}}^{C}$ is even more curious, as the initial increase in flexibility allows for higher UU, but adding more flexibility can even cause the UU values to decrease. Also, notice that the UU values are quite small. This is because a wider use of the support $\widehat{\text{TCIT}}$ would require most crisp realisations for the processing times to be extreme values in the support of each TFN representing processing times, which would contradict the possibility distribution given by each TFN.



(**a**) Random solutions UU (**b**) Searched solutions UU

Figure 7. Average $UU(\widehat{\text{TCIT}})$ values (y-axis) obtained for all instances (x-axis) using $\widehat{\text{TCIT}}^{N}$ (in black), $\widehat{\text{TCIT}}^{K}$ (in dark grey), and $\widehat{\text{TCIT}}^{C}$ (in light grey). Instances with the same size and resource dependency in processing times are grouped so that increasing flexibility levels are joined via a line.

In addition to the above assessment, we can also evaluate the crisp TCIT that is obtained with the schedules obtained using the tabu search with each model for computing $\widehat{\text{TCIT}}$. The objective is to assess the TCIT that can be obtained according to simulations, rather than using a central tendency measure such as the expected value, which, as we have seen, can be deviated due to the artificial uncertainty that is introduced. Figure 8 shows the average executed TCIT across the 1000 crisp scenarios and the 100 different schedules.

These results show a completely different behaviour with respect to the one observed when comparing based on the fuzzy values. It seems that, according to simulations, in those instances with less flexibility, $\widehat{\text{TCIT}}^{N}$ seems to work better than $\widehat{\text{TCIT}}^{K}$; however, as flexibility increases, making the problems harder, $\widehat{\text{TCIT}}^{K}$ takes the lead. There is no clear explanation for this behaviour other than that instances with less flexibility allow to make very few moves between resources, so most moves in the tabu search are limited

to reordering of operations in one resource, and then having less artificial uncertainty in the idle times does not suppose a big advantage, as there would be few chances to find an operation that fills the gap. Meanwhile, $\widehat{\text{TCIT}}^{N}$ involves higher artificial uncertainty that skews TFNs more to the right, which means a higher deviation towards the worst case, which, for this set of instances, seems to be a beneficial trait in the guiding function for lower-flexibility cases. In any case, $\widehat{\text{TCIT}}^{C}$ remains the dominating alternative and the better approach when the problem at hand does not require computing idle times at the operation level.

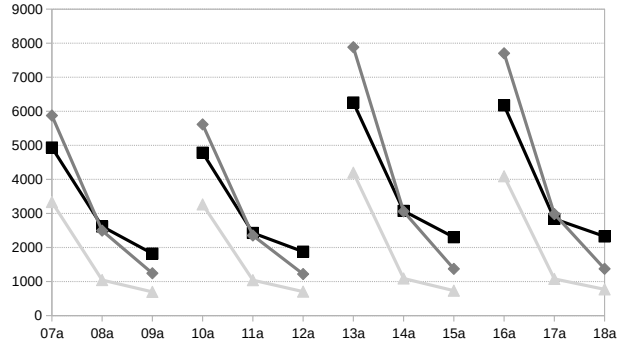


Figure 8. Average executed TCIT values (y-axis) obtained for all instances (x-axis) using $\widehat{\text{TCIT}}^{N}$ (in black), $\widehat{\text{TCIT}}^{K}$ (in dark grey) and $\widehat{\text{TCIT}}^{C}$ (in light grey). Instances with the same size and resource dependency in processing times are grouped so that increasing flexibility levels are joined via a line.

5. Conclusions and Future Work

In this work, we have considered the fuzzy flexible job shop scheduling problem, with a focus on computing idle times. We have illustrated the difficulties inherent to computing idle times in the fuzzy framework, caused by operating with interactive uncertain variables. Starting from a naive definition of idle time before an operation from the literature, we have proposed an alternative way to compute such idle time that takes into account existing knowledge about schedules. These two means of computing idle times before operations yield two different ways of computing the core idle time of a resource and, by extension, the total core idle time of a schedule. Additionally, we have proposed a way of directly computing the core idle time in a resource (and, hence, the total core idle time of a schedule) for those cases where not that much granularity is necessary. We have argued that, although, in the deterministic setting, all three of these approaches are equivalent and yield the same value for the total core idle time in a schedule, that is not the case in the fuzzy setting. We finally conducted an experimental study to empirically evaluate the three ways of computing idle times in fuzzy schedules. We have seen how both the new proposals from this work seem superior to the naive means of calculating idle times that exist in the literature. On the other hand, neither of the two new proposals can be thought of as being superior to the other one. Moreover, the choice of which to use is heavily dependent on the particular application context. The knowledge-based approach from Section 3.2 allows fine-grained calculations at the operation level, whereas the coarser approach from Section 3.3 is adequate if only calculations at the resource level are needed.

We hope that our proposals provide a solid basis for future research tackling fuzzy flexible job shop problems where idle times play an important role. Additionally, it would be trivial to extend our new approach to computing waiting times for jobs or to adapt them to computing both idle and waiting times in other fuzzy scheduling problems. In particular, being able to compute idle times between operations in fuzzy schedules is crucial

to tackling all of those problems that comprise time-of-use tariffs, i.e., problems in which the offer of energy or its cost varies over time.

Author Contributions: Conceptualization, P.G.G., C.R.V. and I.G.-R.; methodology, P.G.G., C.R.V. and I.G.-R.; software, P.G.G.; validation, P.G.G., C.R.V. and I.G.-R.; formal analysis, P.G.G., C.R.V. and I.G.-R.; investigation, P.G.G.; data curation, P.G.G.; writing—original draft preparation, P.G.G.; writing—review and editing, C.R.V. and I.G.-R.; visualization, P.G.G.; supervision, C.R.V. and I.G.-R.; funding acquisition, C.R.V. and I.G.-R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Spanish Government with grant numbers TED2021-131938B-I00 and PID2022-141746OB-I00 and by Universidad de Cantabria and the Government of Cantabria under Grant Concepción Arenal UC-20-20.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Detailed Results

Table A1. Detailed results for the $\mathbb{E}[\widehat{\text{TCIT}}]$.

Instance	$\widehat{\text{TCIT}}^N$				$\widehat{\text{TCIT}}^K$				$\widehat{\text{TCIT}}^C$			
	Random		Searched		Random		Searched		Random		Searched	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
07a	46,763	4262	20,247	954	34478	3398	14,014	1157	17,282	2253	3380	484
08a	46,965	3406	14,435	815	34,209	2715	6404	615	17,385	1856	1525	164
09a	47,658	3984	12,777	674	34,624	3161	3819	431	17,729	2012	1251	112
10a	45,956	4125	19,562	1006	33,999	3320	13,302	1018	17,059	2199	3330	563
11a	45,682	3667	14,029	827	33,267	2882	5932	600	17,194	1855	1530	211
12a	46,672	3831	12,931	711	33,887	3027	3817	374	17,410	1975	1264	95
13a	66,132	5088	25,379	1348	48,515	3961	17,009	1337	24,335	2491	4260	608
14a	65,062	6131	18,453	978	47,218	4750	7839	703	23,803	3067	1755	164
15a	65,772	5340	17,420	1179	47,912	4136	4047	567	24,491	2643	1485	98
16a	65,485	5313	25,418	1186	48,093	4103	17,335	1300	24,024	2475	4156	645
17a	64,342	6410	18,452	831	46,713	5004	7955	759	23,434	2983	1769	160
18a	64,947	4812	17,077	1072	47,316	3748	3938	538	24,136	2578	1550	110

Table A2. Detailed results for the spread.

Instance	$\widehat{\text{TCIT}}^N$				$\widehat{\text{TCIT}}^K$				$\widehat{\text{TCIT}}^C$			
	Random		Searched		Random		Searched		Random		Searched	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
07a	147,838	12,314	71,084	3354	98,697	9006	43,995	3360	11,161	548	7159	544
08a	148,281	9852	52,516	3083	97,256	7462	20,676	1892	11,077	475	4587	266
09a	150,121	11,516	47,423	2777	97,983	8569	12,827	1418	11,162	522	4156	194
10a	144,567	11,711	68,639	3368	96,738	8625	41,679	3044	11,219	545	7129	617
11a	143,276	11,097	51,291	3106	93,616	8234	19,043	1921	10,778	495	4580	334
12a	146,896	11,070	47,933	2950	95,753	8188	12,874	1207	10,909	491	4198	198
13a	209,340	15,214	88,931	4530	138,871	11,104	51,855	3756	14,622	683	8788	725
14a	205,675	17,720	67,640	3858	134,298	12,541	25,274	2036	14,493	768	5620	276
15a	207,036	16,895	65,027	4837	135,595	12,571	13,478	1837	14,759	779	5194	182
16a	207,284	16,048	89,251	3902	137,716	11,501	53,483	4094	14,702	712	8764	798
17a	203,567	19,024	68,111	3237	133,053	13,766	25,875	2390	14,407	822	5680	261
18a	204,513	14,629	63,574	4407	133,990	10,860	12,989	1757	14,550	669	5387	213

Table A3. Detailed results for the MVP.

Instance	$\widehat{\text{TCIT}}^N$				$\widehat{\text{TCIT}}^K$				$\widehat{\text{TCIT}}^C$			
	Random		Searched		Random		Searched		Random		Searched	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
07a	−0.805	0.016	−0.873	0.014	−0.707	0.027	−0.759	0.029	−0.093	0.014	−0.200	0.070
08a	−0.805	0.017	−0.902	0.013	−0.701	0.030	−0.770	0.040	−0.088	0.017	−0.674	0.071
09a	−0.805	0.015	−0.923	0.012	−0.700	0.028	−0.812	0.042	−0.100	0.018	−0.798	0.060
10a	−0.806	0.015	−0.874	0.015	−0.710	0.025	−0.759	0.026	−0.080	0.015	−0.208	0.080
11a	−0.802	0.018	−0.908	0.015	−0.697	0.032	−0.766	0.045	−0.096	0.016	−0.670	0.090
12a	−0.804	0.014	−0.921	0.012	−0.699	0.027	−0.816	0.045	−0.094	0.016	−0.797	0.048
13a	−0.805	0.014	−0.871	0.013	−0.706	0.025	−0.726	0.031	−0.097	0.017	−0.221	0.077
14a	−0.809	0.015	−0.909	0.013	−0.707	0.026	−0.766	0.041	−0.090	0.017	−0.753	0.063
15a	−0.804	0.016	−0.928	0.011	−0.700	0.031	−0.801	0.047	−0.096	0.017	−0.857	0.046
16a	−0.807	0.013	−0.872	0.014	−0.709	0.023	−0.740	0.023	−0.093	0.014	−0.251	0.082
17a	−0.811	0.014	−0.917	0.011	−0.710	0.026	−0.776	0.040	−0.096	0.016	−0.756	0.059
18a	−0.805	0.016	−0.925	0.011	−0.702	0.030	−0.789	0.053	−0.093	0.014	−0.850	0.049

Table A4. Detailed results for the $RDEV$.

Instance	$\widehat{\text{TCIT}}^N$				$\widehat{\text{TCIT}}^K$				$\widehat{\text{TCIT}}^C$			
	Random		Searched		Random		Searched		Random		Searched	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
07a	−0.625	0.024	−0.756	0.024	−0.492	0.033	−0.582	0.039	0.016	0.014	−0.018	0.052
08a	−0.625	0.023	−0.819	0.023	−0.484	0.034	−0.612	0.056	0.015	0.015	−0.321	0.088
09a	−0.622	0.021	−0.857	0.022	−0.480	0.032	−0.676	0.062	0.017	0.014	−0.449	0.099
10a	−0.625	0.023	−0.756	0.025	−0.492	0.030	−0.579	0.038	0.014	0.014	−0.026	0.055
11a	−0.618	0.024	−0.827	0.025	−0.475	0.036	−0.605	0.061	0.016	0.014	−0.331	0.104
12a	−0.622	0.021	−0.854	0.021	−0.478	0.032	−0.682	0.062	0.016	0.014	−0.446	0.091
13a	−0.626	0.019	−0.754	0.021	−0.491	0.029	−0.538	0.040	0.016	0.013	−0.020	0.054
14a	−0.629	0.022	−0.833	0.022	−0.489	0.032	−0.612	0.057	0.015	0.014	−0.384	0.093
15a	−0.622	0.023	−0.867	0.019	−0.481	0.035	−0.663	0.068	0.016	0.013	−0.510	0.094
16a	−0.628	0.018	−0.757	0.021	−0.493	0.026	−0.556	0.031	0.016	0.013	−0.021	0.056
17a	−0.631	0.020	−0.846	0.019	−0.491	0.031	−0.627	0.054	0.016	0.013	−0.397	0.092
18a	−0.623	0.022	−0.863	0.019	−0.482	0.034	−0.653	0.074	0.016	0.013	−0.507	0.092

Table A5. Detailed results for the UU (%).

Instance	$\widehat{\text{TCIT}}^N$				$\widehat{\text{TCIT}}^K$				$\widehat{\text{TCIT}}^C$			
	Random		Searched		Random		Searched		Random		Searched	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
07a	0.81	0.10	0.90	0.12	1.22	0.16	1.53	0.20	10.75	1.20	8.72	1.29
08a	0.83	0.11	1.01	0.15	1.26	0.18	2.52	0.36	11.04	1.46	10.49	1.45
09a	0.79	0.09	1.01	0.14	1.22	0.15	3.63	0.52	10.67	1.21	10.48	1.39
10a	0.82	0.11	0.91	0.11	1.23	0.17	1.66	0.23	10.54	1.39	8.66	1.19
11a	0.81	0.10	1.00	0.12	1.24	0.16	2.73	0.40	10.75	1.25	10.10	1.36
12a	0.78	0.09	1.01	0.14	1.20	0.14	3.48	0.51	10.53	1.17	10.48	1.61
13a	0.75	0.10	0.87	0.11	1.13	0.16	1.66	0.22	10.72	1.40	8.93	1.37
14a	0.75	0.09	0.92	0.11	1.14	0.14	2.66	0.46	10.56	1.26	10.21	1.29
15a	0.77	0.10	0.87	0.13	1.17	0.16	4.05	0.66	10.71	1.25	9.90	1.26
16a	0.75	0.10	0.86	0.12	1.13	0.15	1.57	0.19	10.53	1.25	8.79	1.22
17a	0.74	0.10	0.92	0.13	1.13	0.16	2.60	0.34	10.40	1.33	9.62	1.34
18a	0.75	0.10	0.92	0.14	1.15	0.16	4.02	0.66	10.57	1.31	9.65	1.33

Table A6. Detailed results for the crisp values.

Instance	$\widehat{\text{TCIT}}^{N}$				$\widehat{\text{TCIT}}^{K}$				$\widehat{\text{TCIT}}^{C}$			
	Random		Searched		Random		Searched		Random		Searched	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
07a	17,559	2283	4936	574	17,559	2283	5876	853	17,559	2283	3335	594
08a	17,650	1877	2618	369	17,650	1877	2495	488	17,650	1877	1044	227
09a	18,028	2051	1821	261	18,028	2051	1243	304	18,028	2051	696	172
10a	17,297	2213	4785	609	17,297	2213	5615	737	17,297	2213	3265	679
11a	17,470	1881	2428	382	17,470	1881	2352	471	17,470	1881	1041	289
12a	17,693	2007	1878	241	17,693	2007	1219	301	17,693	2007	704	147
13a	24,722	2525	6258	713	24,722	2525	7884	1063	24,722	2525	4197	766
14a	24,162	3094	3074	402	24,162	3094	3058	626	24,162	3094	1091	239
15a	24,877	2685	2302	285	24,877	2685	1372	371	24,877	2685	732	172
16a	24,399	2513	6174	699	24,399	2513	7704	782	24,399	2513	4092	811
17a	23,808	3015	2841	358	23,808	3015	2983	582	23,808	3015	1077	242
18a	24,508	2609	2325	283	24,508	2609	1374	388	24,508	2609	771	180

Appendix B. Pseudocode

```
tabu_search(solution: Solution, neighborhood: Neighborhood, tabu_list_min_size_lb: Int,
tabu_list_min_size_ub: Int, tabu_list_max_size_lb: Int, tabu_list_max_size_ub: Int, max_iter: Int) ->
Solution

tabu_list : TabuList = TabuList::new(
   Int::random(tabu_list_min_size_lb,tabu_list_min_size_ub),
   Int::random(tabu_list_max_size_lb,tabu_list_max_size_ub)
   )
best_solution : Solution = solution
no_impr_it : Int = 0
while no_impr_it <  max_iter
   no_impr_it += 1
   selected_neighbor : Neighbor = null
   neighbors : MinHeap<Neighbor> = MinHeap::new(neighborhood.neighbors(solution), n =>
n.solution().value())
   while !neighbors.empty()
      neighbor : Neighbor = neighbors.pop()
      if !tabu_list.is_tabu(neighbor.movement()) || neighbor.solution().value() < best_solution.value()
         selected_neighbor = neighbor.solution()
         break
   if selected_neighbor == null
      tabu_list.clear()
      tabu_list.resize(
         Int::random(tabu_list_min_size_lb, tabu_list_min_size_ub),
         Int::random(tabu_list_max_size_lb, tabu_list_max_size_ub)
         )
      continue
   if selected_neighbor.solution().value() < best_solution.value()
      best_solution = neighbor.solution()
      no_impr_it = 0
      tabu_list.clear()
   tabu_list.add(neighbor.movement())
   solution = selected_neighbor.solution()
return best_solution
```

Figure A1. Tabu search pseudocode.

References

1. Baker, K.; Trietsch, D. *Principles of Sequencing and Scheduling*, 2nd ed.; Wiley Series in Operations Research and Management Science; Wiley: Hoboken, NJ, USA, 2019.
2. Pinedo, M.L. *Scheduling. Theory, Algorithms, and Systems*, 6th ed.; Springer: Berlin/Heidelberg, Germany, 2022. [CrossRef]
3. Corsini, R.R.; Costa, A.; Fichera, S.; Parrinello, V. Hybrid Harmony Search for Stochastic Scheduling of Chemotherapy Outpatient Appointments. *Algorithms* **2022**, *15*, 424. [CrossRef]
4. Ghafari, R.; Kabutarkhani, F.H.; Mansouri, N. Task scheduling algorithms for energy optimization in cloud environment: A comprehensive review. *Clust. Comput.* **2022**, *25*, 1035–1093. [CrossRef]
5. Guo, W.; Xu, P.; Zhao, Z.; Wang, L.; Zhu, L.; Wu, Q. Scheduling for airport baggage transport vehicles based on diversity enhancement genetic algorithm. *Nat. Comput.* **2020**, *19*, 663–672. [CrossRef]
6. Tarazona-Torres, L.; Amaya, C.; Paipilla, A.; Gomez, C.; Alvarez-Martinez, D. The Parallel Machine Scheduling Problem with Different Speeds and Release Times in the Ore Hauling Operation. *Algorithms* **2024**, *17*, 348. [CrossRef]

7. Teoh, C.K.; Wibowo, A.; Ngadiman, M.S. Review of state of the art for metaheuristic techniques in Academic Scheduling Problems. *Artif. Intell. Rev.* **2015**, *44*, 1–21. [CrossRef]
8. Xiong, H.; Shi, S.; Ren, D.; Hu, J. A survey of job shop scheduling problem: The types and models. *Comput. Oper. Res.* **2022**, *142*, 105731. [CrossRef]
9. Xie, J.; Gao, L.; Peng, K.; Li, X.; Li, H. Review on flexible job shop scheduling. *IET Collab. Intell. Manuf.* **2019**, *1*, 67–77. [CrossRef]
10. Dauzère-Pérès, S.; Ding, J.; Shen, L.; Tamssaouet, K. The flexible job shop scheduling problem: A review. *Eur. J. Oper. Res.* **2024**, *314*, 409–432. [CrossRef]
11. Chen, J.C.; Wu, C.C.; Chen, C.W.; Chen, K.H. Flexible job shop scheduling with parallel machines using Genetic Algorithm and Grouping Genetic Algorithm. *Expert Syst. Appl.* **2012**, *39*, 10016–10021. [CrossRef]
12. Alvarez-Valdes, R.; Fuertes, A.; Tamarit, J.; Giménez, G.; Ramos, R. A heuristic to schedule flexible job-shop in a glass factory. *Eur. J. Oper. Res.* **2005**, *165*, 525–534. [CrossRef]
13. Bakon, K.; Holczinger, T.; Sule, Z.; Jasko, S. Scheduling Under Uncertainty for Industry 4.0 and 5.0. *IEEE Access* **2022**, *10*, 74977–75017. [CrossRef]
14. Prade, H. Using fuzzy set theory in a scheduling problem: A case study. *Fuzzy Sets Syst.* **1979**, *2*, 153–165. [CrossRef]
15. Chanas, S.; Kamburowski, J. The use of fuzzy variables in pert. *Fuzzy Sets Syst.* **1981**, *5*, 11–19. [CrossRef]
16. Dubois, D.; Fargier, H.; Fortemps, P. Fuzzy Scheduling: Modelling flexible constraints vs. coping with incomplete knowledge. *Eur. J. Oper. Res.* **2003**, *147*, 231–252. [CrossRef]
17. Azadegan, A.; Porobic, L.; Ghazinoory, S.; Samouei, P.; Kheirkha, A.S. Fuzzy logic in manufacturing: A review of literature and a specialized application. *Int. J. Prod. Econ.* **2011**, *132*, 258–270. [CrossRef]
18. Alburaikan, A.; Garg, H.; Khalifa, H.A.E.W. A Novel Approach for Minimizing Processing Times of Three-Stage Flow Shop Scheduling Problems Under Fuzziness. *Symmetry* **2023**, *15*, 130. [CrossRef]
19. Ptuskin, A.; Levner, E.; Kats, V. Cyclic multi-hoist scheduling with fuzzy processing times in flexible manufacturing lines. *Appl. Soft Comput.* **2024**, *165*, 112014. [CrossRef]
20. Abdullah, S.; Abdolrazzagh-Nezhad, M. Fuzzy Job-Shop Scheduling Problems: A Review. *Inf. Sci.* **2014**, *278*, 380–407. [CrossRef]
21. Behnamian, J. Survey on fuzzy shop scheduling. *Fuzzy Optim. Decis. Mak.* **2016**, *15*, 331–366. [CrossRef]
22. Campo, E.A.; Cano, J.A.; Gómez-Montoya, R.; Rodríguez-Velásquez, E.; Cortés, P. Flexible Job Shop Scheduling Problem with Fuzzy Times and Due-Windows: Minimizing Weighted Tardiness and Earliness Using Genetic Algorithms. *Algorithms* **2022**, *15*, 334. [CrossRef]
23. Abdel-Basset, M.; Mohamed, R.; El-Shahat, D.; Sallam, K. An efficient hybrid optimization method for Fuzzy Flexible Job-Shop Scheduling Problem: Steady-state performance and analysis. *Eng. Appl. Artif. Intell.* **2023**, *123*, 106249. [CrossRef]
24. Li, R.; Gong, W.; Lu, C. Self-adaptive multi-objective evolutionary algorithm for flexible job shop scheduling with fuzzy processing time. *Comput. Ind. Eng.* **2022**, *168*, 108099. [CrossRef]
25. García Gómez, P.; González-Rodríguez, I.; Vela, C.R. Enhanced memetic search for reducing energy consumption in fuzzy flexible job shops. *Integr. Comput.-Aided Eng.* **2023**, *30*, 151–167. [CrossRef]
26. García Gómez, P.; Vela, C.R.; González-Rodríguez, I. Neighbourhood search for energy minimisation in flexible job shops under fuzziness. *Nat. Comput.* **2023**, *22*, 685–704. [CrossRef]
27. Pan, Z.; Lei, D.; Wang, L. A Bi-Population Evolutionary Algorithm with Feedback for Energy-Efficient Fuzzy Flexible Job Shop Scheduling. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *52*, 5295–5307. [CrossRef]
28. Shi, D.; Zhang, B.; Li, Y. A Multi-Objective Flexible Job-Shop Scheduling Model Based on Fuzzy Theory and Immune Genetic Algorithm. *Int. J. Simul. Model.* **2020**, *19*, 123–133. [CrossRef]
29. Chen, X.l.; Li, J.Q.; Du, Y. A hybrid evolutionary immune algorithm for fuzzy flexible job shop scheduling problem with variable processing speeds. *Expert Syst. Appl.* **2023**, *233*, 120891. [CrossRef]
30. Wang, Z.; Liao, W.; Zhang, Y. Rescheduling optimisation of sustainable multi-objective fuzzy flexible job shop under uncertain environment. *Int. J. Prod. Res.* **2024**, *62*, 8904–8920. [CrossRef]
31. Gen, M.; Lin, L.; Ohwada, H. Advances in Hybrid Evolutionary Algorithms for Fuzzy Flexible Job-shop Scheduling: State-of-the-Art Survey. In Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021), Online, 4–6 February 2021; Volume 1, pp. 562–573. [CrossRef]
32. Maassen, K.; Perez-Gonzalez, P.; Günther, L.C. Relationship between common objective functions, idle time and waiting time in permutation flow shop scheduling. *Comput. Oper. Res.* **2020**, *121*, 104965. [CrossRef]
33. Alfieri, A.; Garraffa, M.; Pastore, E.; Salassa, F. Permutation flowshop problems minimizing core waiting time and core idle time. *Comput. Ind. Eng.* **2023**, *176*, 108983. [CrossRef]
34. de Abreu, A.P.; Fuchigami, H.Y. An efficiency and robustness analysis of warm-start mathematical models for idle and waiting times optimization in the flow shop. *Comput. Ind. Eng.* **2022**, *166*, 107976. [CrossRef]
35. Park, M.J.; Ham, A. Energy-aware flexible job shop scheduling under time-of-use pricing. *Int. J. Prod. Econ.* **2022**, *248*, 108507. [CrossRef]

36. Gao, K.; Huang, Y.; Sadollah, A.; Wang, L. A review of energy-efficient scheduling in intelligent production systems. *Complex Intell. Syst.* **2020**, *6*, 237–249. [CrossRef]
37. Meng, L.; Zhang, C.; Shao, X.; Ren, Y. MILP models for energy-aware flexible job shop scheduling problem. *J. Clean. Prod.* **2019**, *210*, 710–723. [CrossRef]
38. Li, H.; Zhu, H.; Jiang, T. Modified Migrating Birds Optimization for Energy-Aware Flexible Job Shop Scheduling Problem. *Algorithms* **2020**, *13*, 44. [CrossRef]
39. Duan, J.; Wang, J. Energy-efficient scheduling for a flexible job shop with machine breakdowns considering machine idle time arrangement and machine speed level selection. *Comput. Ind. Eng.* **2021**, *161*, 107677. [CrossRef]
40. Benedikt, O.; Aliko¸, B.; Šůcha, P.; Čelikovský, S.; Hanzálek, Z. A polynomial-time scheduling approach to minimise idle energy consumption: An application to an industrial furnace. *Comput. Oper. Res.* **2021**, *128*, 105167. [CrossRef]
41. Yang, X.W.; Wu, X.C.; Shao, Y.; Tang, G. Energy-delay-aware VNF scheduling: A reinforcement learning approach with hierarchical reward enhancement. *Clust. Comput.* **2024**, *27*, 7657–7671. [CrossRef]
42. Babor, M.; Pedersen, L.; Kidmose, U.; Paquet-Durand, O.; Hitzmann, B. Application of Non-Dominated Sorting Genetic Algorithm (NSGA-II) to Increase the Efficiency of Bakery Production: A Case Study. *Processes* **2022**, *10*, 1623. [CrossRef]
43. Zhao, J.; Peng, S.; Li, T.; Lv, S.; Li, M.; Zhang, H. Energy-aware fuzzy job-shop scheduling for engine remanufacturing at the multi-machine level. *Front. Mech. Eng.* **2019**, *14*, 474–488. [CrossRef]
44. González-Rodríguez, I.; Puente, J.; Palacios, J.J.; Vela, C.R. Multi-objective evolutionary algorithm for solving energy-aware fuzzy job shop problems. *Soft Comput.* **2020**, *24*, 16291–16302. [CrossRef]
45. Afşar, S.; Palacios, J.J.; Puente, J.; Vela, C.R.; González-Rodríguez, I. Multi-objective enhanced memetic algorithm for green job shop scheduling with uncertain times. *Swarm Evol. Comput.* **2022**, *68*, 101016. [CrossRef]
46. Dubois, D.; Prade, H. Fuzzy Numbers: An Overview. In *Readings in Fuzzy Sets for Intelligent Systems*; Dubois, D., Prade, H., Yager, R.R., Eds.; Morgan Kaufmann: Burlington, MA, USA, 1993; pp. 112–148. [CrossRef]
47. Palacios, J.J.; González-Rodríguez, I.; Vela, C.R.; Puente, J. Coevolutionary makespan optimisation through different ranking methods for the fuzzy flexible job shop. *Fuzzy Sets Syst.* **2015**, *278*, 81–97. [CrossRef]
48. Palacios, J.J.; Puente, J.; Vela, C.R.; González-Rodríguez, I. Benchmarks for fuzzy job shop problems. *Inf. Sci.* **2016**, *329*, 736–752. [CrossRef]
49. González Rodríguez, I.; Puente, J.; Vela, C.R.; Varela, R. Semantics of Schedules for the Fuzzy Job Shop Problem. *IEEE Trans. Syst. Man Cybern. Part A* **2008**, *38*, 655–666. [CrossRef]
50. Fortemps, P. Jobshop Scheduling with Imprecise Durations: A Fuzzy Approach. *IEEE Trans. Fuzzy Syst.* **1997**, *7*, 557–569. [CrossRef]
51. Dell' Amico, M.; Trubian, M. Applying Tabu Search to the Job-shop Scheduling Problem. *Ann. Oper. Res.* **1993**, *41*, 231–252. [CrossRef]