Complex & Intelligent Systems
https://doi.org/10.1007/s40747-021-00461-3

ORIGINAL ARTICLE

A neutrosophic set-based TLBO algorithm for the flexible job-shop scheduling problem with routing flexibility and uncertain processing times

Liangliang Jin[1] · Chaoyong Zhang[2] · Xiaoyu Wen[3] · Chengda Sun[1] · Xinjiang Fei[1]

Received: 19 February 2021 / Accepted: 3 July 2021

Abstract
Different with the plain flexible job-shop scheduling problem (FJSP), the FJSP with routing flexibility is more complex and it can be deemed as the integrated process planning and (job shop) scheduling (IPPS) problem, where the process planning and the job shop scheduling two important functions are considered as a whole and optimized simultaneously to utilize the flexibility in a flexible manufacturing system. Although, many novel meta-heuristics have been introduced to address this problem and corresponding fruitful results have been observed; the dilemma in real-life applications of resultant scheduling schemes stems from the uncertainty or the nondeterminacy in processing times, since the uncertainty in processing times will disturb the predefined scheduling scheme by influencing unfinished operations. As a result, the performance of the manufacturing system will also be deteriorated. Nevertheless, research on such issue has seldom been considered before. This research focuses on the modeling and optimization method of the IPPS problem with uncertain processing times. The neutrosophic set is first introduced to model uncertain processing times. Due to the complexity in the math model, we developed an improved teaching-learning-based optimization(TLBO) algorithm to capture more robust scheduling schemes. In the proposed optimization method, the score values of the uncertain completion times on each machine are compared and optimized to obtain the most promising solution. Distinct levels of fluctuations or uncertainties on processing times are defined in testing the well-known Kim's benchmark instances. The performance of computational results is analyzed and competitive solutions with smaller score values are obtained. Computational results show that more robust scheduling schemes with corresponding neutrosophic Gantt charts can be obtained; in general, the results of the improved TLBO algorithm suggested in this research are better than those of other algorithms with smaller score function values. The proposed method in this research gives ideas or clues for scheduling problems with uncertain processing times.

Keywords Neutrosophic sets · Flexible job-shop scheduling · Processing planning · Teaching-learning-based optimization, TLBO · Integrated process planning and scheduling, IPPS · Uncertain processing times

Introduction

As two crucial components in a flexible manufacturing system, the job-shop scheduling module and the process planning module received a number of research attentions [1–5]. In tradition, process planning specifies the technical details [6,7], e.g., cutting parameters; the scheduling module, on the other hand, arranges operations on machines to shorten the maximum completion time (makespan) or meet other criteria [8,9]. These two modules usually perform separately and sequentially [10–12]. Nevertheless, such paradigm ignores the inherent relationship between the two functions, since there is a lack of coordination mechanism between them and more importantly, the flexibility in the two functions

✉ Liangliang Jin
jll2009@foxmail.com

Chaoyong Zhang
zcyhust@hust.edu.cn

[1] Department of Mechanical Engineering, Shaoxing University, Shaoxing, China

[2] School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, China

[3] Henan Key Lab of Intelligent Manufacturing of Mechanical Equipment, Zhengzhou University of Light Industry, Zhengzhou, China

Published online: 17 July 2021

cannot be utilized fully [13,14]. For example, the processing flexibility is rigidified, since one cannot change processing sequence at the scheduling stage with sequential paradigm. Thus, this will lead to inefficiencies of a flexible manufacturing system [15,16], such as resource conflicts and unbalanced utilization of machines [17]. Lots of efforts have been paid to eliminate resource conflicts and improve the performance of a manufacturing system by taking the advantage of the flexibilities in the two modules [4,8,13,17]. Fruitful results have been observed and the meta-heuristics have garnered wide research interests due to the high cost performance and the accessibility.

There are three kinds of flexibilities in this problem: operation flexibility (OF), sequencing flexibility (SF), and processing flexibility (PF) [14,18]. OF means that an operation may be processed by more than one machine tools. SF allows more than one feasible operation permutations as long as these operations satisfy the precedence constraints. PF means that there may be more than one possible operation combinations (or sets) to finish the same feature of a part [10], since the same feature can usually be realized by many feasible operation combinations. Clearly, it is quite necessary to carry out joint optimization for the problem by properly managing OF, SF, and PF.

Many efforts have been paid for the plain IPPS problem. Nevertheless, these research papers mainly pay attentions to the optimization methods to shorten the makespan. In other literature, researchers either consider the problem in a dynamic environment with random job arrivals or performing multi-objective optimizations. Most of these studies deem the processing times as static ones; however, this goes against with real-life situations: the processing times always vary in a certain range due to various kinds of disturbances [19]. As a result, the makespan will fluctuations within a certain range [20]. Clearly, existing optimization techniques for deterministic processing times are not fit for the uncertain IPPS problem any more [21]: the actual starting times or finishing times will always deviate largely from the predetermined ones, and there will be large deviations between the actual makespan and the so-called "optimal" one. Unfortunately, modeling and optimization techniques on the uncertain IPPS problem have seldom been considered and robust scheduling schemes are not available to absorb processing time varyings or hedge against the uncertainty in processing times. Real-life requirements stress the need to perform this study.

There are mainly two types of methods to handle uncertain scheduling problems: the on-line mode and the off-line mode. They also correspond to the reactive scheduling method and the proactive (or preventive) scheduling method [21,22]. In reactive scheduling paradigm, an optimal scheduling scheme is first generated; when disturbances occur or starting times of operations deviate from the predetermined values to a certain extent, the rescheduling process is trigged in time by the scheduling module for the remaining operations. Such rescheduling procedure will be performed interactively till all the operation have been processed. This paradigm cannot ensure the global optimality, and the total makespam will increase with the number of reschedulings [23,24]. The proactive scheduling considers the possible processing time fluctuations or disturbances in advance and the resultant scheduling scheme is capable to absorb possible processing time fluctuations. In this research, the proactive scheduling paradigm is adopted and we try to obtain a scheduling scheme with certain immunity to uncertain processing times.

Several methods have been proposed to model actual processing times. For example, random variables that subject to a certain probability distribution have been considered [25]. In other cases, processing times are treated as fuzzy numbers that subject to a certain kind of membership function [26–28], such as triangular fuzzy numbers (TFNs) or trapezoidal fuzzy numbers (TrFNs). Usually, it is very hard to distinguish which distribution operation processing times follow and the math operators. The fuzzy set-based optimization techniques have received more research attentions due to its convenience in modeling uncertain processing times as well as in the implementation details of algorithms. Traditional fuzzy theory is not perfect in describing the fuzziness of things. Prof. Smarandache further developed the concept of "neutrosophy" in 2008 for a better conveying of people's thinking [29]. After that, the neutrosophic set was formally proposed to efficiently and effectively cope with indeterminate and inconsistent information [30,31]; as a powerful general framework which generalizes classic sets, fuzzy sets, intuitionistic fuzzy sets, tautological sets, and other sets [32], the neutrosophic set is capable to handle incomplete information, indeterminate information, and inconsistent information [32]. Therefore, it can be used to more precisely or accurately describe uncertain objects.

In fuzzy sets, there is membership function μ_x only and the information of indeterminacy and nonmembership is lost [32]. In neutrosophic sets, indeterminacy is quantified explicitly as three independent components: the truth membership $(T_A(x))$, the indeterminacy membership $(I_A(x))$, and the falsity membership $(F_A(x))$. Because all the three memberships are used to reflect the ambiguous nature of subjective judgments, there is no restriction to express uncertain objectives and it is quite easy to capture imprecise or uncertain information. Nevertheless, since the three components $T_A(x)$, $I_A(x)$, and $F_A(x)$ are non-standard subsets (non-standard interval $]0^-,\ 1^+[$), they cannot be directly used in engineering problems or real-world applications. Therefore, the single-valued neutrosophic set (SVNS), a branch of neutrosophic sets, is proposed [33,34]. Let E be a universe; each of the three membership functions of an SVNS is mapped into the closed interval: $T_A : E \rightarrow [0, 1]$, $I_A : E \rightarrow [0, 1]$, $F_A : E \rightarrow [0, 1]$. Successful applications

of SVNSs have been reported. Deli et al. developed a ranking method for single-valued neutrosophic numbers and they applied the method in multi-attribute decision-making [35]. Pramanik and Mallick developed a TODIM strategy to deal with multi-attribute group decision-making problem, where score function, accuracy function, and Hamming distance function for single-valued trapezoidal neutrosophic numbers were considered [36]. The shortest path problem in neutrosophic set environment has also been considered [37–40]. For example, Broumi et al. in their research suggested a new score function for interval-valued neutrosophic numbers and the neutrosophic shortest path is determined based on the score function [39]; the score function is used to evaluate the paths that have been chosen. Applications of SVNSs can also be found in pattern recognition and medical diagnosis [41,42], taxonomy, and clustering analysis [42] and other areas. Unfortunately, most of the existing studies regarding SVNSs or related applications focus on decision-making issues; to the best of our knowledge, there is no research regarding SVNSs for the uncertain IPPS problem. In this study, SVNSs are first introduced to model the uncertain processing times in solving the IPPS problem.

Scheduling problems, due to their complexity and the NP-hardness, are usually solved by meta-heuristic algorithms. Inspired by the effects of influence of teachers on learners, Rao et al. in 2008 proposed the teaching-learning-based optimization algorithm [43]. The TLBO algorithm also hires many individuals as learners and teacher(s). Nevertheless, the outstanding feature of TLBO is parameter independent; that is, every learner will take part in the 'teacher phase' and the 'learner phase' and there is no limitations caused by pre-set probabilities throughout the two phases. Relative applications of the TLBO algorithm can be found in existing literature. Rao and Patel used the TLBO algorithm in multi-objective optimization of heat exchangers to achieve maximum heat exchanger effectiveness with minimum total cost [44]; they also give the improved version of the plain TLBO algorithm to enhance the exploration and exploitation capacities [45]. Tang et al. suggested a hybrid TLBO algorithm for solving multi-constraints' stochastic two-sided assembly line balancing problem [46]. In terms of scheduling problems, applications of the TLBO algorithm have also been reported [47–49]; for example, Shao et al. proposed a hybrid discrete TLBO algorithm for the no idle flow shop scheduling problem to minimize the total tardiness [48]. In this research, we extent the application of this algorithm to the discrete problem and an improved TLBO algorithm is developed for the IPPS problem.

This research tries to develop a neutrosophic based TLBO algorithm for the IPPS problem which is contaminated with uncertain processing times. Neutrosophic sets are used to model uncertain processing times; this is the novelty of this research. To improve the exploitation ability of the algorithm, some modifications are made for the TLBO algorithm to adapt to the problem. The remainder of the paper is organized as follows. Some research papers on the IPPS problem as well as the scheduling problem with uncertain processing times will be reviewed in "Literature review". "Mathematical modeling with neutrosophic set" gives some preliminaries on neutrosophic sets; besides, the method to convert a deterministic mathematical model of the IPPS problem to the neutrosophic counter part will also be presented in this section. In the next section, details of the proposed improved TLBO algorithm is demonstrated. The experimental study with discussions will be arranged in "Experiments with discussions". Conclusions with further research directions will be given in the last section.

Literature review

The IPPS problem with deterministic processing times has been investigated widely [10,16,17,50–53]. The research of the IPPS problem starts from single objective optimization, and many research papers mentioned above pay more attentions to makespan reduction. Many novel optimization algorithms have been applied in their optimization methods. For instance, Kim et al. suggested the symbiotic evolutionary algorithm to tackle this problem [17]; Zhang et al. developed an object-coding genetic algorithm to optimize the makespan criterion [50]; Lian et al. introduced the imperialist competitive algorithm in IPPS instance optimizations with relative promising results [10]. Besides, Petrovic et al. and Jin et al. also adopted novel meta-heuristic algorithms to address deterministic IPPS problem [51,54]. Recently, Liu et al. proposed a modified genetic algorithm and the corresponding encoding and decoding methods for the IPPS problem and promising results have been observed [13]. Relative literatures have been summarized in Table 1.

Some researchers have shifted their attentions to the IPPS problems with practical requirements, e.g., handling uncertainty in IPPS instances. As analyzed in "Introduction", the fluctuations in processing times will affect the upcoming operations (they will be put off) and further disturb the whole scheduling plan; if such issue is not settled properly, the original scheduling scheme will be useless. Therefore, it is quite necessary to develop effective optimization method to hedge against such uncertainty. One of the paradigms is the reactive or on-line rescheduling approach [23]; the rescheduling will be triggered when there is a large deviation between the actual and the predefined scheduling scheme. However, the renewed scheduling scheme cannot ensure the global optimality [23]. Therefore, the proactive paradigm received more research attentions. There are some kinds of methods in the proactive scheduling paradigm.

Table 1 Summary of relative literature regarding IPPS optimization

References	Problem	Methodology	Achievements
Kim et al. [17]	Plain IPPS problem	Symbiotic evolutionary algorithm with novel coding scheme	The proposed 24 instances have been optimized
Shao et al. [52]	Plain IPPS problem	Modified genetic algorithm	A new integration model is proposed and some small-scale instances have been optimized
Lian et al. [10]	Plain IPPS problem	Imperialist competitive algorithm	The Kim's benchmark and other instances have been optimized
Li et al. [16]	Multi-objective IPPS problem(makespam, maximal machine workload, total machine workload)	Nash equilibrium based hybrid genetic algorithm	Provide a new idea on the multi-objective IPPS problem
Haddadzade et al. [8]	IPPS problem with uncertain processing times	Dijkstra algorithm, Monte Carlo sampling, and a hybrid SA and Tabu Search(TS) algorithm	Four instances were generated and solved to reflect the robustness of the proposed method
Zhang and Wong [50]	Plain IPPS problem	Object-coding genetic algorithm (OCGA)	Provide a new GA based method in solving the IPPS problem and Kim's benchmarks have been improved
Petrovic et al. [54]	Multi-objective IPPS problem(makespan, balanced level of machine utilization, mean flow time)	Chaos theory based particle swarm optimization (PSO) algorithm	The proposed algorithm outperforms GA, simulated annealing (SA) and hybrid algorithm and relative benchmarks have been optimized
Jin et al. [23]	Multi-objective IPPS in dynamic environment	Periodic and event-driven rescheduling strategies based NSGA-II algorithm	Rescheduling interval, newly added jobs and shop utilization will influence the efficiency of a manufacturing system
Zhang and Wong [53]	Plain IPPS problem	Enhanced ant colony optimization (E-ACO)	The Kim's benchmark has been improved
Li et al. [26]	IPPS problem with uncertain processing times	Combination of PSO and GA based on interval numbers	The hybrid algorithm outperforms GA and is effective in uncertain IPPS optimization
Liu et al. [13]	Plain IPPS problem	Modified genetic algorithm (MGA)	The Kim's benchmark and real-world instances have been optimized

The chance constrained programming (CCP) approach is the first kind of approach that is used in scheduling problems or other discrete optimization problems with parameters uncertainty [55–57]. This method can transform a mixed integer linear programming (MILP) model with uncertain parameters into a "deterministic" one, and the resultant model can resist the uncertainty in parameters. A typical application of such method is reported in [57]; the processing times in a flow shop are regarded as random variables and an equivalent deterministic model is provided. Similar research can also be found in recent publication [58]. Other applications of CCP can be found in assembly line balancing [56] and project scheduling [55], etc. However, this method relies on the deterministic MILP model of the problem; in many cases, e.g., the IPPS problem and the flexible job-shop scheduling problem, the corresponding model is quite complex with lots of variables and constraints and the model after conversion is more complex than before. This hinders the application of the method.

Fuzzy set-related optimization methods are another kind of approach in coping with uncertainty in scheduling problems. Because this method can be easily combined with meta-heuristic algorithms in tackling complex scheduling problems or other NP-hard problems, many fuzzy set-related studies have been investigated since 1999 [59]. Sakawa and Mori gave a textbook like example of such application in job-shop scheduling in the genetic algorithm framework [59]; they adopted triangular fuzzy numbers to model the uncertain processing times and due dates and the approximation for the max operator was developed to ensure that the result is also a triangular fuzzy number. In subsequent studies, many fuzzy set-based optimization methods for parameter uncertain scheduling problems have been published, and meta-heuristic algorithms are adopted. Lei suggested a decomposition–integration genetic algorithm (DIGA) to reduce the fuzzy makespan value for the flexible job-shop scheduling problem where the uncertain processing times are also mapped into triangular fuzzy numbers [27]. Later, they gave a similar research, where an efficient swarm-based neighborhood search algorithm (SNSA) is developed for the fuzzy flexible job-shop scheduling problem [60]. Following Lei's step, Wang et al. combined fuzzy numbers with the artificial bee colony (ABC) algorithm and presented a hybrid artificial bee colony (HABC) algorithm in their research to capture the best fuzzy makespan [61]; again, the uncertain processing times are treated as triangular fuzzy numbers. With almost the same coding and decoding methods, Wang et al. also suggested an effective fuzzy number-based estimation of distribution algorithm (EDA) to obtain the best makespan in the flexible job shop [28]. Gao et al. reported a study on uncertain flexible job shop scheduling problems [1]; a discrete harmony search (DHS) algorithm with a simple heuristic rule which is used to initialize the individuals was proposed to shorten the maximum fuzzy completion time. Li et al. recently presented a research regarding the uncertain IPPS problem based on the interval number [26], which can be deemed as a kind of fuzzy number with a uniform distribution-like membership function. In recent years, multi-objective cases of uncertain scheduling problems have been reported. For example, Gao et al. proposed an improved artificial bee colony (IABC) algorithm to minimize the maximum fuzzy completion time and the maximum fuzzy machine workload, and the benchmark instances as well as the practical instances were adopted to test the algorithm [62]. According to the literature mentioned above, it can be found that the uncertain processing times are regarded as triangular fuzzy numbers and other types of fuzzy numbers have seldom been considered. The reason behind this is that triangular fuzzy numbers are more close to the actual situations. However, existing investigations pay less attention on the uncertain IPPS problem and worse still, there is no application of neutrosophic sets on the uncertain IPPS problem.

Other kinds of methods dealing with uncertain scheduling problems have also been reported according to related literature. For instance, Liu et al. suggested a Petri net-based model for the emergency response process which is constrained by resources and uncertain durations [63]. Haddadzade et al. considered the uncertain IPPS problems using a two stage method, where the process planning procedure and the scheduling module are separated [8]; nevertheless, in their research, only several 'promising' process plans are considered, and hence, the corresponding flexibilities cannot be fully utilized. Different with existing research, we propose in this paper a novel optimization method for the uncertain IPPS problem; the neutrosophic set is applied to model the uncertain processing times. The resultant scheduling scheme is capable to hedge against the uncertainty and improve the robustness at a certain extent.

Mathematical modeling with neutrosophic set

The uncertain IPPS problem

The IPPS problem can be deemed as the extension of the flexible job-shop scheduling problem. However, the process planning module increases the flexibilities as well as complexity of the problem. Based on Ref. [64], the uncertain IPPS problem can be defined as: given a set of n parts (jobs) to be processed on m machines with operations that have alternative manufacturing resources and uncertain processing times, select the suitable manufacturing resources and sequence the operations so as to determine a schedule in which the precedence constraints among operations can be satisfied and the

corresponding objectives, e.g., the fuzzy maximum completion time, can be optimized. In this research, the actual processing time of each operation on each available machine will be presented using neutrosophic sets.

A job in the IPPS problem can be presented by a network graph, as shown in Fig. 1. The starting node and the ending node are the dummy nodes, representing the beginning and the finishing operations. The other two types of nodes are operation nodes and 'OR' nodes; an operation node stands for an operation where the operation ID, alternative machines with corresponding processing times have been specified. For instance, operation 6 in Fig. 1 can be processed on machine 1 or 5 with nominal processing times 42 and 38, respectively. The 'OR' node sometimes may not appear in a network graph; however, if it appears after a certain node, there will be at least two OR link paths, each of which begins after the 'OR' node and ends when the path merges with the other [14]. For example, there are two OR link paths after the 'OR' node among operation nodes 8, 9, and 12 in Fig. 1; therefore, only the left OR link path (operation nodes 9, 10, 11) or the right OR link path (operation nodes 12, 13) needs to be visited. If a bifurcation without 'OR' node, operation nodes in all the link paths should be visited. Operation sets 2, 3 and 4, 5 are in two link paths according to Fig. 1, and these two link paths are not OR link paths; therefore, operation nodes 2–5 have to be visited. The arrows in Fig. 1 indicate the precedence relationships between operations: an one-way arrow from node A to B means that operation B should be processed directly or indirectly after operation A. In such a case, a job may have many possible process plans (operation permutations), since there may be lots of operation precedence relationships specified by the one-way arrows in the network graph. Especially, there are quite a few operation nodes whose precedence relationships are not determined, since no arrow is placed between any two operation nodes, e.g., operation nodes 3 and 12 in Fig. 1. Two feasible process plans (operation permutations) of the example network graph are also given in Fig. 1. With such flexibility, one does not know which operation permutation is the 'best' one from the scheduling point of view; clearly, it is quite necessary to consider both the scheduling and the process planning modules simultaneously: a 'bad' process plan in process planning module may be the best one in scheduling.

The three types of flexibilities are reflected in a network graph. The situation where an operation can be processed by more than one available machines reflects the operation flexibility; the situation where given operations can have different permutations as long as they satisfy the precedence relationships stands for the sequencing flexibility; finally, the situation where only operations in one OR link path are selected relates to the processing flexibility.

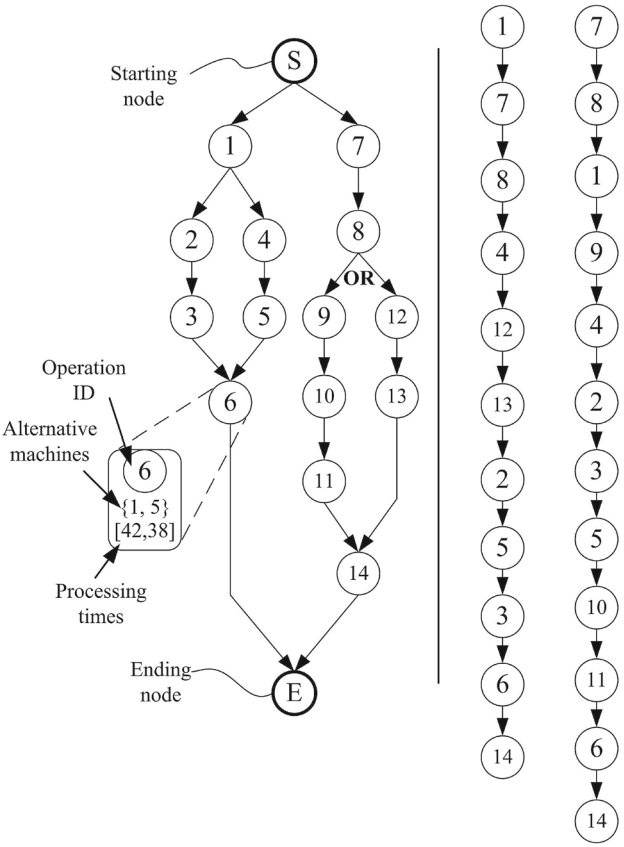

Fig. 1 A network graph with two possible process plans

Neutrosophic sets

This section introduces some basic concepts of neutrosophic sets, and related operations.

Definition 1 [31,40] Let 1 be a special neutrosophic set on the real number set $\mathbb{R}$, the truth membership function $\mu_{\tilde{a}}(x)$, the indeterminacy membership function $\nu_{\tilde{a}}(x)$, and the falsity membership function $\lambda_{\tilde{a}}(x)$ of are defined as

$$\mu_{\tilde{a}}(x) = \begin{cases} \frac{T_{\tilde{a}}(x-\tilde{a}_T)}{\tilde{a}_I-\tilde{a}_T}, & \tilde{a}_T \le x \le \tilde{a}_I, \\ T_{\tilde{a}}, & \tilde{a}_I \le x \le \tilde{a}_P, \\ \frac{T_{\tilde{a}}(\tilde{a}_S-x)}{\tilde{a}_S-\tilde{a}_P}, & \tilde{a}_P \le x \le \tilde{a}_S, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

$$\nu_{\tilde{a}}(x) = \begin{cases} \frac{(\tilde{a}_I-x+I_{\tilde{a}}(x-\tilde{a}_T))}{\tilde{a}_I-\tilde{a}_T}, & \tilde{a}_T \le x \le \tilde{a}_I, \\ I_{\tilde{a}}, & \tilde{a}_I \le x \le \tilde{a}_P, \\ \frac{(x-\tilde{a}_P+I_{\tilde{a}}(\tilde{a}_S-x))}{\tilde{a}_S-\tilde{a}_P}, & \tilde{a}_P \le x \le \tilde{a}_S, \\ 1, & \text{otherwise.} \end{cases} \tag{2}$$

$$\lambda_{\tilde{a}}(x) = \begin{cases} \frac{(\tilde{a}_I-x+F_{\tilde{a}}(x-\tilde{a}_T))}{\tilde{a}_I-\tilde{a}_T}, & \tilde{a}_T \le x \le \tilde{a}_I, \\ F_{\tilde{a}}, & \tilde{a}_I \le x \le \tilde{a}_P, \\ \frac{(x-\tilde{a}_P+F_{\tilde{a}}(\tilde{a}_S-x))}{\tilde{a}_S-\tilde{a}_P}, & \tilde{a}_P \le x \le \tilde{a}_S, \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

For the case when $0 \leq \tilde{a}_T \leq \tilde{a}_I \leq \tilde{a}_P \leq \tilde{a}_S \leq 1$ and $T_{\tilde{a}}, I_{\tilde{a}}, F_{\tilde{a}} \in [0, 1]$, $\tilde{a}$ is called a normalized trapezoidal neutrosophic number; when $\tilde{a}_I = \tilde{a}_P$, $\tilde{a}$ is transformed into a triangular neutrosophic number. In this research, the uncertain processing time are modeled as triangular neutrosophic numbers.

Definition 2 [33,40] Let X be a space point, and x be an element in X, a single valued neutrosophic set(NS) V in X is characterized by membership functions, e.g., $\mu_{\tilde{a}}(x)$, $\nu_{\tilde{a}}(x)$, and $\lambda_{\tilde{a}}(x)$, and $\mu_{\tilde{a}}(x) : X \to [0, 1]$, $\nu_{\tilde{a}}(x) : X \to [0, 1]$, $\lambda_{\tilde{a}}(x) : X \to [0, 1]$.

It is clear that, the truth, the indeterminacy and the falsity membership functions are mapped into the range $[0, 1]$ in single-valued NSs; this brings convenience for the applications of NSs. However, operators of triangular neutrosophic sets (TNSs) are required to be defined, because the operators are used in the scheduling (decoding) procedure. More precisely, three operators of TNSs, e.g., the addition operator, the maximization operator, and the ranking operator, should be ensured. The addition operator is defined to determine the sum of two TNS numbers in completion time calculation. The the ranking operator is employed in comparing two TNSs numbers so as to compare the completion time on each machine, and hence, the makespan is determined. The maximization operator is developed to determine the neutrosophic starting time of an operation. Based on the existing literature [31,40], we have

Definition 3 Let $\tilde{r}^N = \langle[\tilde{r}_T, \tilde{r}_I, \tilde{r}_S], (T_{\tilde{r}}, I_{\tilde{r}}, F_{\tilde{r}})\rangle$ and $\tilde{s}^N = \langle[\tilde{s}_T, \tilde{s}_I, \tilde{s}_S,], (T_{\tilde{s}}, I_{\tilde{s}}, F_{\tilde{s}})\rangle$ be two TNSs, and some operators are defined as:

$$\begin{aligned}\tilde{r}^N \oplus \tilde{s}^N &= \langle[\tilde{r}_T + \tilde{s}_T, \tilde{r}_I + \tilde{s}_I, \tilde{r}_S + \tilde{s}_S], \\ &\quad (T_{\tilde{r}} + T_{\tilde{s}} - T_{\tilde{r}}T_{\tilde{s}}, I_{\tilde{r}}I_{\tilde{s}}, F_{\tilde{r}}F_{\tilde{s}})\rangle . \qquad (4)\\ \tilde{r}^N \otimes \tilde{s}^N &= \langle[\tilde{r}_T \cdot \tilde{s}_T, \tilde{r}_I \cdot \tilde{s}_I, \tilde{r}_S \cdot \tilde{s}_S], \\ &\quad (T_{\tilde{r}} \cdot T_{\tilde{s}}, I_{\tilde{r}} + I_{\tilde{s}} - I_{\tilde{r}}I_{\tilde{s}}, F_{\tilde{r}} + F_{\tilde{s}} - F_{\tilde{r}}F_{\tilde{s}})\rangle . \qquad (5)\\ \theta\tilde{r}^N &= \left\langle[\theta\tilde{r}_T, \theta\tilde{r}_I, \theta\tilde{r}_S], (1 - (1 - T_{\tilde{r}})^\theta, (I_{\tilde{r}})^\theta, (F_{\tilde{r}})^\theta)\right\rangle . \qquad (6)\end{aligned}$$

Definition 4 Let $\tilde{r}^N = \langle[\tilde{r}_T, \tilde{r}_I, \tilde{r}_S], (T_{\tilde{r}}, I_{\tilde{r}}, F_{\tilde{r}})\rangle$ be a TNS, and the score function can be defined as

$$s\left(\tilde{r}^N\right) = \frac{1}{12}\left[\tilde{r}_T + 2\tilde{r}_I + \tilde{r}_S\right] \times [2 + T_{\tilde{r}} - I_{\tilde{r}} - F_{\tilde{r}}] . \qquad (7)$$

Let $\tilde{r}^N = \langle[\tilde{r}_T, \tilde{r}_I, \tilde{r}_S], (T_{\tilde{r}}, I_{\tilde{r}}, F_{\tilde{r}})\rangle$ and $\tilde{s}^N = \langle[\tilde{s}_T, \tilde{s}_I, \tilde{s}_S,], (T_{\tilde{s}}, I_{\tilde{s}}, F_{\tilde{s}})\rangle$ be two TNSs, and the ranking of the two TNSs is easy: if $s\left(\tilde{r}^N\right) \prec s\left(\tilde{s}^N\right)$ then $\tilde{r}^N \prec \tilde{s}^N$. Furthermore, if $s\left(\tilde{r}^N\right) = s\left(\tilde{s}^N\right)$ coincidentally, the accuracy function is applied [35,40]. In many cases, using the score function is enough to compare two TNS numbers.

For the maximization operator, it is used to determine the maximum neutrosophic completion times of two operations: the current operation should be started only after the maximum neutrosophic completion time between the job predecessor and the machine predecessor. Since the indeterminacy and the falsity membership functions are considered, the maximization operator will be discussed in the decoding procedure in later sections.

IPPS modeling

In this research, the TNS-based mathematical model is developed based on previously proposed Type-2 model [14], which is more powerful and general than Type-1 models [65]; nevertheless, it is suitable for plain or crisp number-based processing times only. We therefore try to introduce TNS numbers into the plain mixed integer linear programming (MILP) model, and then to interpret it into the deterministic one. The MILP model with plain processing times is provided here for ease of understanding.

In modeling the problem, we assume that: (1) job preemptions are not allowed; (2) each machine can only process at most one job at any time; (3) at any time, a job can only be processed by at most one machine; (4) all the jobs are available at time zero; (5) the transportation time as well as the set-up time can be included in the nominal processing time. Corresponding sets, parameters, and variables are given as follows.

Subscripts and notations

i, i' Jobs, $1 \leq i, i' \leq |n|$,
j, j' Operations, $1 \leq j, j' \leq |n_i|$,
k, k' Machines,
h Combinations,
O_{ij} The jth operation of the ith job,
O_{ihj} The jth operation of the ith job in the hth operation combination.

Sets and parameters

p_{ijk} The nominal processing time of O_{ij} processed by machine k,
p^N_{ijk} The neutrosophic processing time of O_{ij} processed by machine k, $p^N_{ijk} = \left\langle[p^T_{ijk}, p^I_{ijk}, p^S_{ijk}], (T^p_{ijk}, I^p_{ijk}, F^p_{ijk})\right\rangle$,
R_{ih} The operation set that contains the operation belonging to the hth combination of the i-th job,
$V_{ijj'}$ =1, If there is an arrow from operation node j to j' in the network graph of job i (O_{ij} is to be processed before $O_{ij'}$); =0, otherwise; This parameter expresses the partial precedence relationships among operations,
K_i The set of operation combinations of job i, job i has $|K_i|$ operation sets,
n The set of jobs,
n_i The set that contains all the operations in the network graph of job i; some of the operations may not be selected due to the 'OR' node(s),
M_{ij} The set of alternative machines of O_{ij},
POS_{ij} The pre-ordered set of O_{ij}; it contains all the operations that should be processed before O_{ij},
BOS_{ij} The back-ordered set of O_{ij}; it contains all the operations that should be processed after O_{ij},

$Q_{ijj'}$ =1, O_{ij} should be processed directly or indirectly before $O_{ij'}$; =0, otherwise,
A A very large positive integer.

The parameter $V_{ijj'}$ determines the precedence relationships of a few operations only; only this set of parameters is not enough to determine the precedence relationships of all the selected operations (the operations belonging to the selected combination). Therefore, the parameters $Q_{ijj'}$s, which specify the precedence relationships of all the operations, are constructed (see the algorithm in Ref. [14]). The basic principle of parameters $Q_{ijj'}$s is easy: if the precedence relationship of any two selected operations is determined, the precedence relationships of all the selected operations are determined.

Variables

$C_{\max}$ The nominal makespan value,
$C^N_{\max}$ The neutrosophic makespan value, $C^N_{\max} = \langle[C^T_{\max}, C^I_{\max}, C^S_{\max}], (T^C_{\max}, I^C_{\max}, F^C_{\max})\rangle$,
$W_{iji'j'}$ =1, If operation O_{ij} is processed before $O_{i'j'}$; =0, otherwise,
Y_{ih}=1, If the h-th combination of the i-th job is selected; =0, otherwise,
$Z_{ijj'}$=1, If operation O_{ij} is processed directly or indirectly before operation $O_{ij'}$; =0, otherwise,
X_{ihjk}=1, If operation O_{ihj} is processed on machine k; =0, otherwise,
C_{ihj} The completion time of operation O_{ihj}.
C^N_{ihj} The neutrosophic completion time of operation O_{ihj}; $C^N_{ihj} = \left\langle[C^T_{ihj}, C^I_{ihj}, C^S_{ihj}], (T^C_{ihj}, I^C_{ihj}, F^C_{ihj})\right\rangle$.

Objective

$$\min C_{\max}. \tag{8}$$

Constraints

$$\sum_{h\in K_i} Y_{ih} = 1, \quad \forall i \tag{9}$$

$$\sum_{k\in M_{ij}} X_{ihjk} = Y_{ih}, \quad \forall i \in n, h \in K_i, \forall j \in R_{ih} \tag{10}$$

$$A \cdot Y_{ih} \geq C_{ihj}, \quad \forall i \in n, h \in K_i, \forall j \in R_{ih} \tag{11}$$

$$C_{ihj'} \geq C_{ihj} + \sum_{k'\in M_{ij'}} X_{ihj'k'} p_{ij'k'}, \quad \forall i \in n, h \in K_i, \forall j, j' \in R_{ih},\ j \neq j',\ V_{ijj'} = 1 \tag{12}$$

$$Z_{ijj'} + Z_{ij'j} = 1, \quad \forall i \in n,\ j, j' \in n_i,\ j \neq j', \quad Q_{ijj'} + Q_{ij'j} = 0 \tag{13}$$

$$C_{ihj'} \geq C_{ihj} + \sum_{k'\in M_{ij'}} X_{ihj'k'} p_{ij'k'} - A(1 - Z_{ijj'}), \quad \forall i \in n,\ \forall h \in K_i, \forall j, j' \in R_{ih},\ j \neq j' \tag{14}$$

$$C_{i'h'j'} \geq C_{ihj} + X_{i'h'j'k'} p_{i'j'k'} - A(1 - W_{iji'j'}) - A(2 - X_{ihjk} - X_{i'h'j'k'}) \quad \forall i, i' \in n,\ i \neq i',\ h \in K_i,\ h' \in K_{i'},\ j \in R_{ih},\ j' \in R_{i'h'},\ k, k' \in M_{ij} \cap M_{i'j'},\ k = k' \tag{15}$$

$$C_{ihj} \geq C_{i'h'j'} + X_{ihjk} p_{ijk} - A \cdot W_{iji'j'} - A(2 - X_{ihjk} - X_{i'h'j'k'}) \quad \forall i, i' \in n,\ i \neq i',\ h \in K_i,\ h' \in K_{i'},\ j \in R_{ih},\ j' \in R_{i'h'},\ k, k' \in M_{ij} \cap M_{i'j'},\ k = k' \tag{16}$$

$$C_{\max} \geq C_{ihj}, \quad \forall i \in n, \forall h \in K_i,\ j \in R_{ih}. \tag{17}$$

Equation (8) gives the objective: to minimize the nominal makespam. Constraint set (9) means that each job is forced to select an operation combination which is divided by the 'OR' link paths in the network graph; only all the necessary operations are selected, can the job be completed. Constraint set (10) indicates that for the selected operations in the hth combination, they should be assigned to exactly one machine; otherwise, operations will not be assigned to any machines ($Y_{ih} = 0$). Constraint set (11) further restricts the completion time of unselected operations: if operations belonging to the hth combination are not selected, their completion times are set to zero. Constraint set (12) determines the completion times of two operations that have a precedence relationship specified directly by the network graph. For the two operations that have no precedence relationship ($Q_{ijj'} + Q_{ij'j} = 0$), constraint set (13) is used to determine which operation will be processed before the other. After this, such operations can be scheduled sequentially based on constraint set (14). Constraint sets (15) and (16) schedule two operations on the same machine by determining the precedence relationship of any two operations on the same machine; if the operation is not processed by machine k, the two constraint sets are useless. Finally, constraint set (17) determines the the completion time of each job.

In the following, we try to interpret the current model into the neutrosophic version. Note that for a real number or a integer a, the corresponding neutrosophic version is $\langle[a, a, a](1.0, 0.0, 0.0)\rangle$ and its score function value is

$\frac{1}{12}[a+2a+a]\times[2+1.0-0.0-0.0]=a$; therefore, there is no need to reform the constraint sets that contain binary variables only, e.g., constraint sets (9) and (10). For other constraint sets that contain continuous variables, they should be reformed. The makespan should be calculated and determined using the score function instead of a crisp number $C_{\max}$; the objective function (8) can thus be reformed as

$$\begin{aligned}\min\ & \frac{1}{12}\left[C^T_{\max}+2C^I_{\max}+C^S_{\max}\right]\\ &\times\left[2+T^C_{\max}-I^C_{\max}-F^C_{\max}\right].\end{aligned}\tag{18}$$

The constraint set (11) is interpreted in constraint set (19), where the strict inequality is converted to the inequality of score function values

$$\begin{aligned}&\frac{A}{12}\times(2+1.0+0.0+0.0)\cdot Y_{ih}\\ &\geq\frac{1}{12}\left[C^T_{ihj}+2C^I_{ihj}+C^S_{ihj}\right]\\ &\quad\times\left[2+T^C_{ihj}-I^C_{ihj}-F^C_{ihj}\right].\end{aligned}\tag{19}$$

For constraint set (12), according to Definition 3, it can be reformed as

$$\begin{aligned}C^N_{ihj'}\geq C^N_{ihj}+\sum_{k'\in M_{ij'}}&\left\langle\left[X_{ihj'k'}p^T_{ij'k'},X_{ihj'k'}p^I_{ij'k'},X_{ihj'k'}p^S_{ij'k'}\right],\right.\\ &\left.\left(1-\left(1-T^p_{ij'k'}\right)^{X_{ihj'k'}},\left(I^p_{ij'k'}\right)^{X_{ihj'k'}},\left(F^p_{ij'k'}\right)^{X_{ihj'k'}}\right)\right\rangle\\ &\forall i\in n,\ h\in K_i,\ \forall j,j'\in R_{ih},\ j\neq j',\ V_{ijj'}=1.\end{aligned}\tag{20}$$

However, constraint set (20) contains the non-linear terms, e.g. $1-\left(1-T^p_{ij'k'}\right)^{X_{ihj'k'}}$; besides, the terms $C^N_{ihj'}$ and C^N_{ihj} can be further unfolded. Fortunately, the binary variable $X_{ihj'k'}$ has only two states: suppose operation $O_{ihj'}$ is selected and processed on machine k'; the result of summation is $\left\langle\left[p^T_{ij'k'},p^I_{ij'k'},p^S_{ij'k'}\right],\left(T^p_{ij'k'},I^p_{ij'k'},F^p_{ij'k'}\right)\right\rangle$; otherwise, it yields $\langle[0,0,0],(0.0,1.0,1.0)\rangle$. Furthermore, according to Definition 3, we have

$$\begin{aligned}&\langle[0,0,0],(0.0,1.0,1.0)\rangle+\cdots+\langle[0,0,0],(0.0,1.0,1.0)\rangle\\ &\quad+\left\langle\left[p^T_{ij'k'},p^I_{ij'k'},p^S_{ij'k'}\right],\left(T^p_{ij'k'},I^p_{ij'k'},F^p_{ij'k'}\right)\right\rangle\\ &=\left\langle\left[p^T_{ij'k'},p^I_{ij'k'},p^S_{ij'k'}\right],\left(T^p_{ij'k'},I^p_{ij'k'},F^p_{ij'k'}\right)\right\rangle.\end{aligned}$$

Therefore, the summation operator in constraint set (20) can be considered separately to reduce the complexity of the model; the refined model is given in constraint set (21)

$$\begin{aligned}&\frac{1}{12}\left[C^T_{ihj'}+2C^I_{ihj'}+C^S_{ihj'}\right]\times\left[2+T^C_{ihj'}-I^C_{ihj'}-F^C_{ihj'}\right]\\ &\geq\left(1-X_{ihj'k'}\right)\frac{1}{12}\left[C^T_{ihj}+2C^I_{ihj}+C^S_{ihj}\right]\\ &\quad\times\left[2+T^C_{ihj}-I^C_{ihj}-F^C_{ihj}\right]\\ &\quad+X_{ihj'k'}\frac{1}{12}\left[C^T_{ihj}+p^T_{ij'k'}+2\left(C^I_{ihj}+p^I_{ij'k'}\right)+C^S_{ihj}+p^S_{ij'k'}\right]\\ &\quad\times\left[2+\left(T^C_{ihj}+T^p_{ij'k'}-T^C_{ihj}T^p_{ij'k'}\right)-I^C_{ihj}I^p_{ij'k'}-F^C_{ihj}F^p_{ij'k'}\right]\\ &\quad\forall i\in n,\ h\in K_i,\quad\forall j,j'\in R_{ih},\ j\neq j',\ V_{ijj'}=1;\end{aligned}\tag{21}$$

constraint set (21) adopts binary variable $X_{ihj'k'}$ to distinguish whether operation $O_{ihj'}$ is processed on machine k'; if so, $\left(1-X_{ihj'k'}\right)$ equals 0 and the neutrosophic completion time $\tilde{C}_{ihj'}=\left\langle\left[C^T_{ihj'},C^I_{ihj'},C^S_{ihj'}\right],\left(T^C_{ihj'},I^C_{ihj'},F^C_{ihj'}\right)\right\rangle$ is determined. In other cases, naturally, $\tilde{C}_{ihj'}\geq\tilde{C}_{ihj}$.

With the similar method, other constraint sets (14)–(17) can also be converted into the corresponding constraint sets; nevertheless, the resultant constraint sets are very complex, and more importantly, massive binary variables with constraints hinder the application of the model. For the MILP models of the IPPS problem with crisp real parameters and variables [14], the results cannot be obtained in reasonable time; for the models of uncertain IPPS problem, one cannot obtain the optimal solutions also, since the model proposed above is more complex than the previous ones [14]. Instead, we try to solve this problem using the soft computing approach.

TLBO-based algorithm

The improved TLBO algorithm

By far, there are many nature-inspired meta-heuristic algorithms; among these algorithms, the genetic algorithm is the most classical algorithm. GA, which hires the Darwin's 'the survival of the fittest', can provide promising solutions for many optimization problems. Nevertheless, the critical failing of GA stems from the parameter setting: the values of key parameters in GA, e.g., the crossover probability, will affect the effectiveness of the solutions. For other meta-heuristics, the determination of parameters is also very empirical. For example, in particle swarm optimization (PSO) algorithm, some parameters, the inertia weight for example, should also be properly determined. In view of this, Rao et al. paid effort in developing novel parameter-free meta-heuristics and they proposed the TLBO algorithm, which simulates the efforts of the influence of teachers on the students.

The TLBO algorithm is also developed based on the swarm intelligence mechanism; in the algorithm, the most knowledgeable individual in the population is mimicked as the teacher and other individuals are the learners. The teacher tries his best to disseminate knowledge to the learners to improve the knowledge levels of the population. Obviously, the knowledge level of the teacher will influence the learners; in many times, learners may require a more superior, qualified, or sophisticated teacher to teach them; besides, the learner may surpass their teacher. In such a case, a new teacher will be selected among the latest population. On the other hand, acquiring knowledge can also be realized by learning from each other; that is, the learners can also improve their knowledge level by consulting other advanced learners. Therefore, the TLBO algorithm consists of two phases: the teacher phase and the learner phase, and there is no probability-related parameter setting requirement in both the two phases.

However, the TLBO algorithm is originally designed for unconstrained non-linear continuous optimization problems; meanwhile, there is only one teacher from whom students can learn. Therefore, the algorithm should be improved properly to adapt to the uncertain IPPS problem which is a discrete optimization problem. In this research, the original TLBO algorithm is properly improved by employing the coding and the decoding scheme discussed below. Moreover, different teachers may be good at different subjects and only one teacher in the algorithm is usually not enough to improve the students' knowledge level. There has the practical significance: the algorithm will be trapped into local optimum with only one teacher in the IPPS problem optimization, since the diversity of the population is largely limited. Therefore, top 5% of the individuals are deemed as the teachers in the improved TIBO algorithm.

Encoding and decoding

In the proposed improved TLBO algorithm, an individual stands for a solution and a solution can also be mapped into a individual; this can be realized using the coding and the decoding procedures. Like previous research [21,23,51], the coding scheme, presented in Fig. 2a, considers operation combination selection, machine selection, and operation permutation simultaneously. The coding scheme contains three strings: the scheduling string, the process plan string, and the operation string. The process plan string contains only one position in which the operation combination ID is recorded. By properly sequencing the operations belonging to the selected operation combination and selecting the corresponding machines, a feasible process plan can be obtained. The operation string contains the information of each operations belonging to the job. Each job has exactly one process plan string and operation string; therefore, the number of jobs, e.g. $|n|$, corresponds to the number of process plan strings as well as operation strings. According to Fig. 2, the process plan string is attached to the corresponding operation string. In an operation string, the sequence of each operations is determined properly using the binary tree method [66] according to the precedence relationships specified in the network graph. The number of positions in the operation string is exactly the number of operations of the selected operation combination. In each position, the selected machine together with the nominal processing time is given in the pair of brackets. However, the number of positions of an operation string depends on the number of operations of the selected operation combination, and it equals $|R_{ih}|$. The scheduling string contains $\sum |R_{ih}|_{\max}$ positions and it is a permutation of job IDs; if the actual number of operations in selected operation combination is less than the maximum one, $|R_{ih}| < |R_{ih}|_{\max}$, the vacant positions will be filled with 0s. In the scheduling string, the operation-based coding paradigm is adopted to avoid any possible infeasibility: if job ID i appears exactly j times in current position, it means that the current operation is located in the jth position of operation string of job i.

For example, there are three jobs in the IPPS instance in Fig. 2a, they contain three, two, and four operations, respectively. The third operation combination is selected in the first job, and there are three operations in this combination. For all the three jobs, there are totally nine operations; thus, there are 9 nonzero positions in the scheduling string. The third position of the scheduling string is number 3 and this number appears exactly for the second time: the third operation to be scheduled is in the second position of the operation string of job 3. This operation is the second operation of job 3 and it will be processed by machine 2.

The decoding procedure is developed based on the active scheduling paradigm in which the insertions of operations are allowed to shorten the makespan. The final makespan is a TNS number; therefore, the decoding procedure will be developed based on the existing deterministic parameter-based scheduling method with some improvements. In the following, the decoding procedure proposed in this research are described.

1. Based on the coding scheme, especially the the scheduling string and the operation string, determine the scheduling sequence according to which the operations will be allocated on machines one by one.
2. For each operation to be scheduled, determine the machine as well as the neutrosophic processing times $\tilde{t}_{ijk} = \left\langle \left[t^T_{ijk}, t^I_{ijk}, t^S_{ijk}\right], \left(T^t_{ijk}, I^t_{ijk}, F^t_{ijk}\right)\right\rangle$.
3. If current operation O_{ijk} is to be scheduled on the machine on which previous operations have located, idle time slots are required to be checked one by one. This process can further be divided into four situations below:

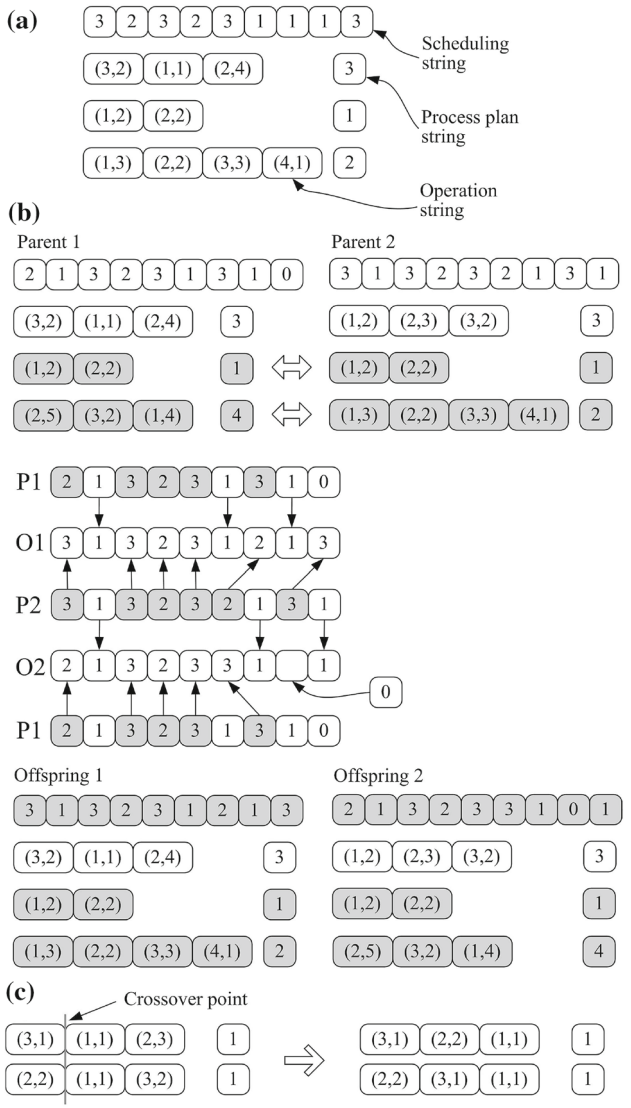

Fig. 2 Coding scheme with crossover operator

3.1 If there is no job predecessor (JP) or machine predecessor (MP), the starting time is regarded as $\langle[0, 0, 0], (0.0, 1.0, 1.0)\rangle$. Calculate the score function (SF) value of the slot end time (SET): $SF(SET)$ which is the starting time of the slot end operation. Calculate the SF value of the current operation end time (COET): $SF(\langle[0, 0, 0], (0.0, 1.0, 1.0)\rangle + \left\langle\left[t_{ijk}^T, t_{ijk}^I, t_{ijk}^S\right], \left(T_{ijk}^t, I_{ijk}^t, F_{ijk}^t\right)\right\rangle) = SF(COET)$. If $SF(COET) \leq SF(SET)$, insert current operation into this time slot. This procedure is illustrated in Fig. 3a, where the curves representing indeterminacy and falsity are not given. In the Gantt chart with neutrosophic processing times, the curves of starting times are given below horizontal lines, while the neutrosophic completion times are above horizontal lines.

3.2 If the current operation has JP and has no MP, the starting time of the time slot depends on the completion time of JP: $\tilde{t}_{JP}$. If $SF(\tilde{t}_{JP} + \tilde{t}_{ijk}) \leq SF(SET)$, insert current operation into this time slot. This procedure is illustrated in Fig. 3b.

3.3 If the current operation has MP and has no JP, the starting time of the time slot depends on the completion time of MP: $\tilde{t}_{MP}$. If $SF(\tilde{t}_{MP} + \tilde{t}_{ijk}) \leq SF(SET)$, insert current operation into this time slot. This procedure is illustrated in Fig. 3c.

3.4 If the current operation has both the MP and JP, the starting time of the time slot depends on the maximum completion time between JP and MP: $\max\left\{\tilde{t}_{JP}, \tilde{t}_{MP}\right\}$. The starting time of current operation can be determined as $\langle t^T = \max\left\{t_{JP}^T, t_{MP}^T\right\}, t^I = \max\left\{t_{JP}^I, t_{MP}^I\right\}, t^S = \max\left\{t_{JP}^S, t_{MP}^S\right\}\rangle$. If $SF\left(\left\langle\left[t^T, t^I, t^S\right], \left(T_{JP}^t, I_{JP}^t, F_{JP}^t\right)\right\rangle\right) \leq SF\left(\left\langle\left[t^T, t^I, t^S\right], \left(T_{MP}^t, I_{MP}^t, F_{MP}^t\right)\right\rangle\right) \leq SF(SET)$, the slot starting time is $\left\langle\left[t_{JP}^T, t_{JP}^I, t_{JP}^S\right], \left(T_{MP}^t, I_{MP}^t, F_{MP}^t\right)\right\rangle$; otherwise, if $SF\left(\left\langle\left[t^T, t^I, t^S\right], \left(T_{MP}^t, I_{MP}^t, F_{MP}^t\right)\right\rangle\right) \leq SF\left(\left\langle\left[t^T, t^I, t^S\right], \left(T_{JP}^t, I_{JP}^t, F_{JP}^t\right)\right\rangle\right) \leq SF(SET)$, the slot starting time is $\left\langle\left[t_{JP}^T, t_{JP}^I, t_{JP}^S\right], \left(T_{JP}^t, I_{JP}^t, F_{JP}^t\right)\right\rangle$. If either of the two cases holds, insert current operation into this time slot. This procedure is illustrated in Fig. 3d.

4. If the current operation cannot be arranged at any idle time slot on a certain machine, it can only be appended at the bottom of the machine. In such a case, as the situation in Step 3.4, the starting time of the current operation is determined by the maximum completion time between its JP and MP.
5. Return to Step 2 to schedule the next operation till all the operations have been processed.

Genetic operators

The crossover procedure is the main genetic operator, and it is responsible for the learning process. In GA, the excellent gene fragments in parents are passed on to their offsprings through the crossover procedure; similarly, the teacher will disseminate knowledge to the learners in TLBO and this is also realized by performing the crossover between the teacher and the learners (the teacher will not be changed after the crossover procedure). As shown in Fig. 2b, c, there are two levels in the crossover operator. For two individuals, e.g., parent 1 (P1) and parent 2 (P2) in figure, determine several jobs randomly. Exchange the operation strings and the process plan strings of the selected jobs, and keep other strings as they are. In Fig. 2b, jobs 2 and 3 are selected and the corresponding operation strings and process plan strings are exchanged. Depending on the selected operation combina-

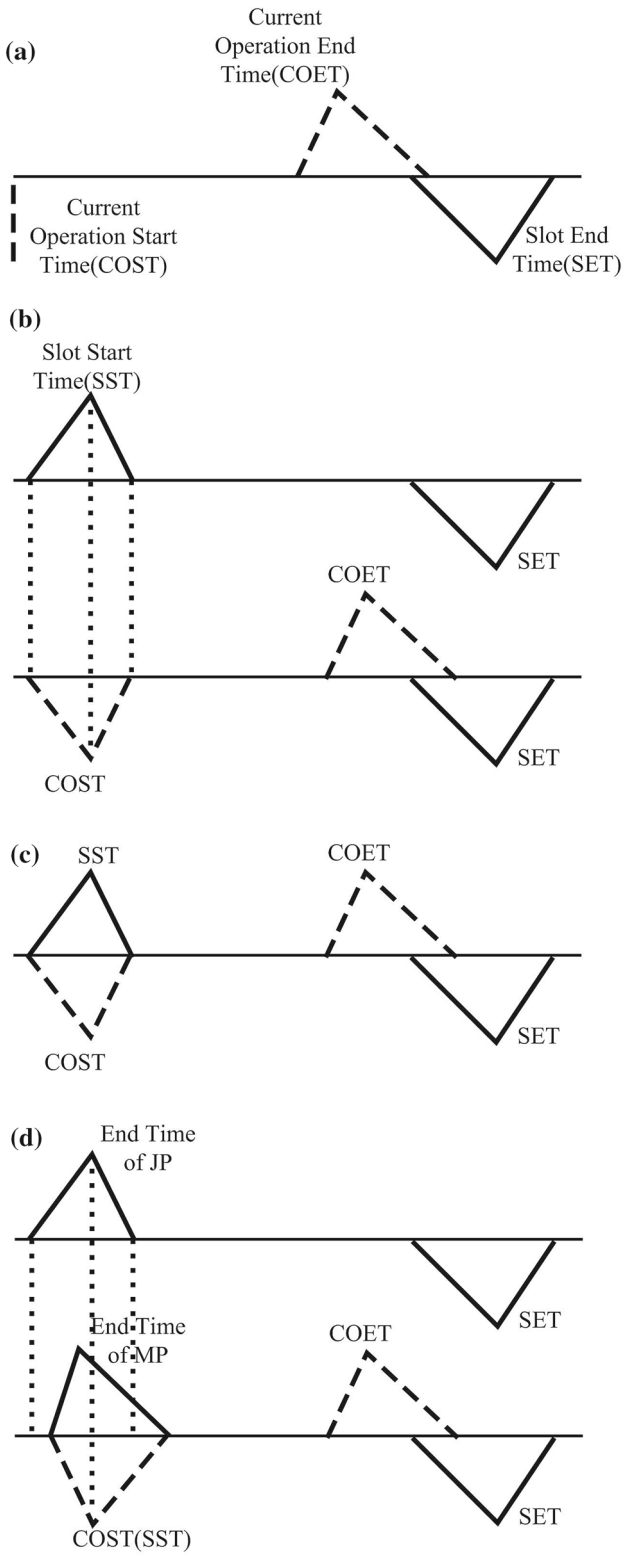

Fig. 3 Decoding scheme

tion, the amount of operations needed to complete a job may be distinct; therefore, this will affect the job IDs in the scheduling strings in both individuals. As presented in Fig. 2b, the scheduling strings in both P1 and P2 are adjusted accordingly: the job IDs of unselected jobs in P1(P2) are kept in the same position in O1(O2), and the job IDs of the selected jobs in P2(P1) are placed at the void positions in offsprings O1(O2) with the same sequence as they are originally in the scheduling string in P2(P1). Finally, the void position(s) will be filled with 0s to avoid any possible infeasibility. In this case, operation combination 4 of job 3 contains only three operations and there will be 8 operations only in O2; thus, a 0 is added in O2. The resultant two offsprings, O1 and O2, are also given in Fig. 2b. In 'teacher' phase, P1 and P2 stand for the teacher and the learner, respectively, and the teacher is recovered as it is before the crossover process. In 'learner' phase, P1 and P2 are two randomly selected individuals.

There is another kind of crossover operator: the crossover in process planning level. The operations in the same operation combination may have different permutations as long as the precedence relationships in the network graph are satisfied. For the operations of the same job in P1 and P2, if they have the identical operation combination (have the same number in the corresponding process plan strings), the crossover operator between operation strings can be performed. The single point crossover [67] is adopted to maintain the feasibility of the two process plan strings. For instance, the first operation combination has been selected by the same job and the single point crossover is performed. A crossover point is randomly determined; the operations before the crossover point keep untouched and the positions after the crossover point are filled with the remainder operations in the other operation string. As a result, two distinct process plans are generated in Fig. 2c. The work flow of the whole algorithm is depicted in Fig. 4.

Experiments with discussions

To simulate the real-life situations, three scenarios, e.g., large uncertainty, medium uncertainty, and small uncertainty, are designed. According to Fig. 5a, some parameters are defined to describe the three scenarios: the fluctuation $|\alpha|$, the range of the truth membership function T, the range of the indeterminacy membership function I, and the range of the falsity membership function F. By properly setting these parameters, these three scenarios can be simulated. For example, the large uncertainty scenario corresponds to larger $|\alpha|$, I, F values and smaller T value, while small $|\alpha|$, I, F values mean the uncertainty has been reduced. The $|\alpha|$ value determines the upper bounds of the t^T, t^S values according to Fig. 5a and the t^I is the nominal processing time of an operation. Table 2 gives the range of these parameters in different scenarios. It can be found out that the t^T, t^S values are uniformly sampled from the range $\left[(1-\alpha)\,t^I, t^I\right]$ and $\left[t^I, (1+\alpha)\,t^I\right]$ respectively; furthermore, the maximum truth and indeter-

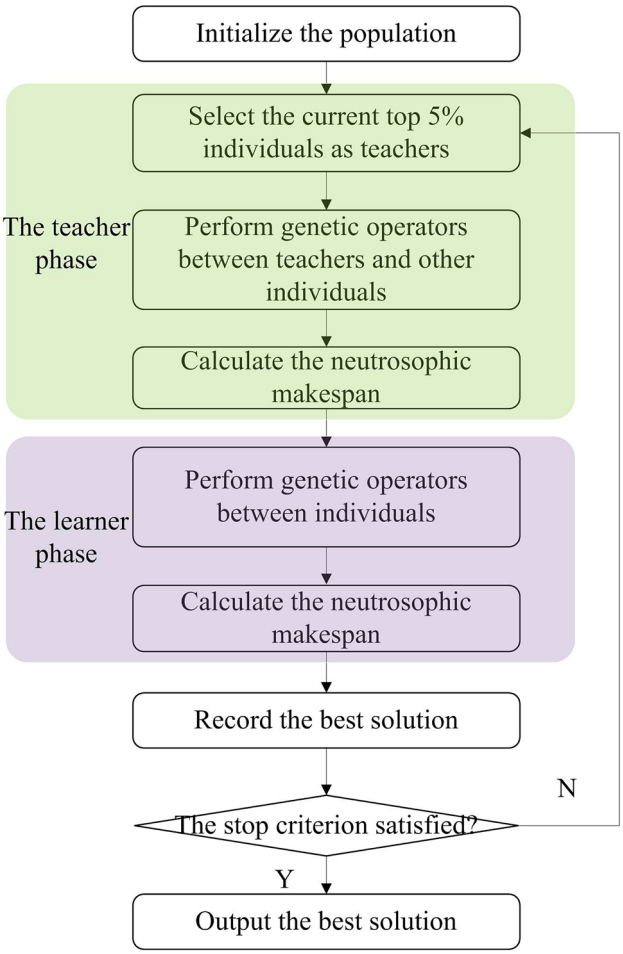

Fig. 4 The work flow of the algorithm

Table 2 Parameter setting in three scenarios

Small uncertainty	Medium uncertainty	Large uncertainty
$\|\alpha\| = 5\%$	$\|\alpha\| = 10\%$	$\|\alpha\| = 15\%$
$T \in [0.75, 1.0]$	$T \in [0.5, 0.75]$	$T \in [0.25, 0.5]$
$I \in [0.0, 0.25]$	$I \in [0.25, 0.5]$	$I \in [0.5, 1.0]$
$F = 1.0 - T$	$F = 1.0 - T$	$F = 1.0 - T$

minacy membership values (the amplitude values) are also uniformly sampled in the given ranges. The maximum falsity membership value can be calculated as $F = 1.0 - T$. Figure 5b gives an example of the actual neutrosophic processing time. For instance, if the truth and the indeterminacy membership values are 0.62 and 0.31 in the medium uncertainty case, the falsity membership value is $1 - 0.62 = 0.38$.

The kim's benchmark [17], which contain 24 instances, is adopted to test the proposed TLBO algorithm. All the 24 instances covers the small-scale, the medium-scale, and the large-scale instances; in each instance, there are 15 machines to process the operations. The objective is to minimize the maximum completion time (score value) as mentioned in objective function (18). The proposed algorithm is coded in C++ language and is performed on a computer with an Intel i5-9600 3.7GHz CPU and 16GB memory. The number of individuals is set to 800 and the algorithm will be stopped after 800 iterations. For each instance, 5 independent computations are performed.

We first give the intuitive presentations on the convergence of the algorithm. The convergence curves of both the score value and the nominal makespan of Instance 24 with large uncertainty (the last instance is the most complex one) in Figs. 6 and 7. Since the score value of the makespan is calculated using the truth, the indeterminacy, and the falsity membership values using objective function (18), it is usually less than the nominal makespan value. With teaching and learning optimization processes going on, both the two values decrease; this reflects the effectiveness of the improved TLBO algorithm. According to Fig. 6, the curve of nominal makespan rises sometimes, while the other curve declines all the way; reasons behind this are: 1) a scheduling scheme with a large nominal makespan value may perform better in resisting the uncertain in processing times and 2) the objective is guided by the score value of the makespan instead of the nominal makespam. It is noteworthy that the nominal makespan value according to Figs. 6 and 7 is larger than that of deterministic cases [50,51]. However, this is reasonable, because compact operation permutations on machines in a scheduling scheme with a small makespan value usually extrude the idle time between operations, and hence, the scheduling scheme cannot absorb processing time fluctuations. In other words, a scheduling scheme with a large makespan value may be much more robust. The similar phenomena can be observed in Fig. 7, where the plain TLBO algorithm is adopted. Nevertheless, the resultant score value of neutrosophic makespan is larger than the case in Fig. 6 (compare the red curves in both the figures). This indicates that the improved TLBO algorithm performs better than the plain TLBO algorithm. Later, the performances of the two algorithms will be further compared.

In the following, the results obtained by both the two algorithms are compared and corresponding results are listed in Tables 3, 4, 5, 6, 7, and 8. The results obtained by the improved TLBO algorithm and the plain TLBO algorithm with large uncertainty in processing times are first compared. For the nominal makespan values, the plain TLBO algorithm performs better than the improved TLBO algorithm for most of the 24 instances. It seems that the plain TLBO algorithm is more promising; however, the nominal makespan value is useless in uncertain IPPS problems, since the actual scheduling scheme may lose effectiveness in real-life cases where the processing times are undetermined. The score value of neutrosophic makespan is more important; as can be seen in the table, score values of neutrosophic makespan of the

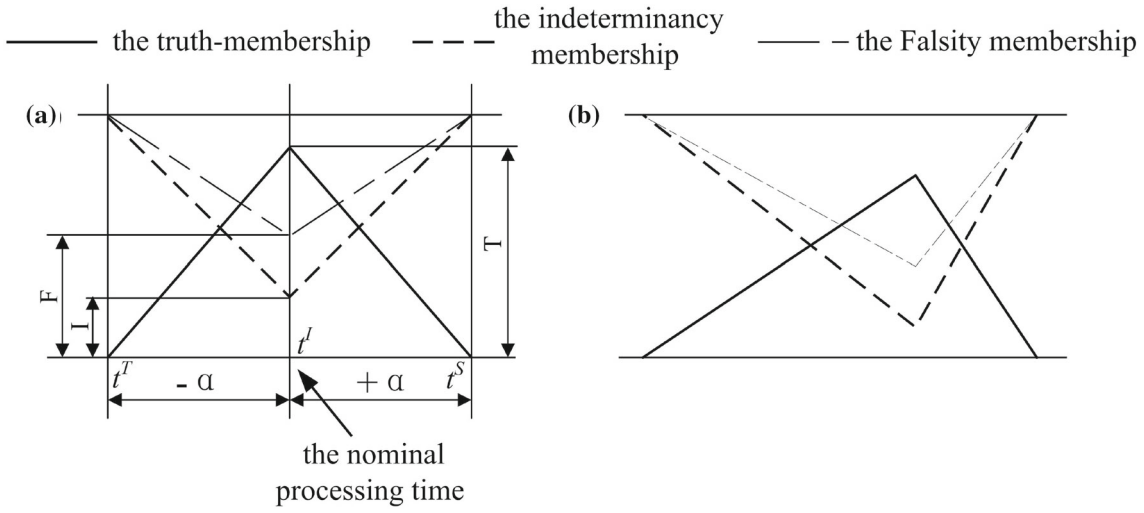

Fig. 5 The graph representation of membership functions of TNS

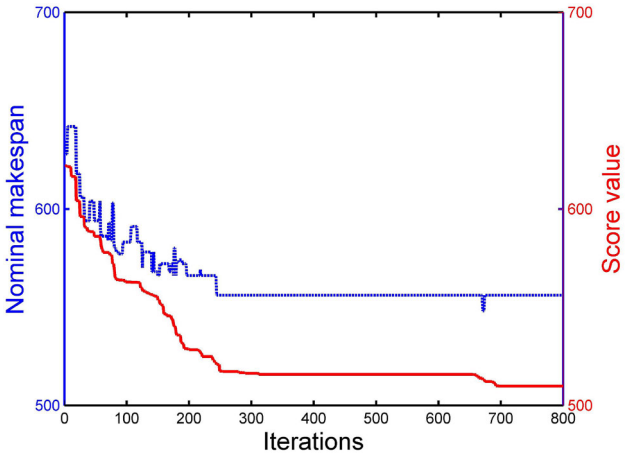

Fig. 6 The convergence curves of Instance 24 with improved TLBO algorithm

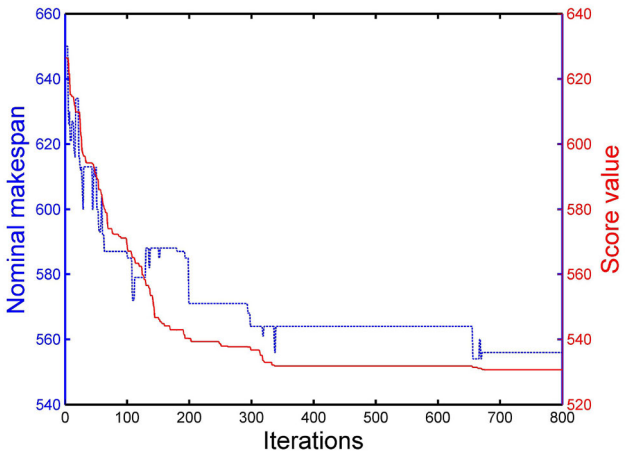

Fig. 7 The convergence curves of Instance 24 with plain TLBO algorithm

improved TLBO algorithm outperform those of plain TLBO algorithm: the improved TLBO algorithm outperform the plain TLBO algorithm for 23 instances among all the 24 instances, and corresponding results in Table 3 are shown in bold. A smaller score value of neutrosophic makespan means that the resultant scheduling scheme is more robust with processing times contaminated with uncertainties. The enhanced 'teacher' phase of the proposed improved TLBO algorithm renders the high diversity of individuals during the learning process, and hence, more competitive results can be observed. Besides, the maximum and the minimum score values of the neutrosophic makespan of the improved TLBO algorithm are also better than those of plain TLBO algorithm; this further testifies that the proposed algorithm is better than the plain TLBO algorithm in uncertain IPPS optimization. As analyzed in the paragraph above, a scheduling scheme with a large nominal makespan value may performed better in uncertain processing time scenarios, since the idle time slots can absorb the uncertainty or fluctuations in processing times; it can be perceived that this conclusion holds after comparing the average nominal makespan values and the average score values obtained by the two algorithms in Table 3. For example, the mean nominal makespan value of Instance 1 of the improved TLBO algorithm is 486.20 and it is larger than the one obtained by the plain TLBO algorithm (458.20); but the average score value of this instance is 392.63 with improved TLBO, and it is better than the one of plain TLBO.

Other meta-heuristic algorithms includng GA, differential evolution (DE) algorithm, and particle swarm optimization (PSO) algorithm have been applied also in this research for comparisons. In PSO algorithm, particles (individuals) follow the best particle as well as the the historical optimal solution of itself. In both the GA and DE algorithms,

Table 3 Results comparisons with large uncertainty

Cases	The improved TLBO		The plain TLBO	
	NM[a] (max, min, mean)	SV[b] (max, min, mean)	NM (max, min, mean)	SV (max, min, mean)
1	(509,465,486.20)	(422.73,373.23,**392.63**)	(469,450,458.20)	(427.68,378.18,401.03)
2	(389,362,373.40)	(323.52,308.16,**316.29**)	(377,355,366.60)	(342.54,313.31,329.28)
3	(421,360,376.40)	(348.03,326.70,**341.55**)	(384,359,368.60)	(352.44,338.74,346.57)
4	(328,319,322.80)	(306.90,269.36,**295.30**)	(336,308,320.20)	(314.82,278.24,302.06)
5	(363,330,350.40)	(318.28,277.25,**307.03**)	(365,321,336.60)	(322.42,307.30,315.56)
6	(526,470,494.20)	(387.09,343.53,**354.87**)	(523,477,496.60)	(387.72,350.27,371.93)
7	(402,376,387.40)	(373.95,315.81,**333.35**)	(392,377,381.80)	(370.26,357.39,367.04)
8	(392,357,370.80)	(339.57,315.46,**324.81**)	(379,354,372.00)	(342.54,318.50,330.76)
9	(509,466,491.80)	(340.56,297.00,**316.16**)	(524,476,497.60)	(345.51,340.56,342.71)
10	(506,452,475.20)	(430.65,389.25,415.53)	(488,459,472.20)	(433.62,369.27,**408.08**)
11	(416,366,384.00)	(365.07,342.72,**352.43**)	(403,366,379.80)	(383.13,337.59,358.18)
12	(352,333,339.40)	(327.69,313.68,**322.38**)	(350,330,338.60)	(336.30,319.68,325.43)
13	(569,482,514.80)	(428.67,360.79,**393.51**)	(531,479,501.60)	(449.67,388.08,421.58)
14	(460,387,421.80)	(378.70,337.59,**362.64**)	(461,415,431.20)	(399.96,377.19,388.44)
15	(524,478,505.40)	(362.34,345.33,**352.40**)	(548,467,508.60)	(391.38,341.55,373.30)
16	(488,462,479.80)	(437.58,398.97,**422.99**)	(516,456,480.00)	(454.41,403.92,427.07)
17	(446,401,422.80)	(380.81,350.46,**366.46**)	(481,428,443.20)	(408.87,393.03,401.56)
18	(406,364,380.20)	(346.50,324.03,**333.42**)	(426,370,389.40)	(379.17,324.47,352.17)
19	(516,471,497.00)	(446.49,396.58,**427.75**)	(557,484,510.80)	(438.57,419.76,427.88)
20	(495,423,448.00)	(400.95,370.26,**389.64**)	(491,430,450.20)	(431.64,392.45,411.65)
21	(536,485,508.60)	(404.91,378.03,**391.81**)	(522,457,490.40)	(436.59,407.88,428.08)
22	(578,524,545.20)	(439.13,409.05,**424.40**)	(562,504,524.20)	(470.42,433.75,461.04)
23	(507,437,470.20)	(422.73,401.66,**414.43**)	(474,439,458.60)	(455.43,413.53,428.27)
24	(559,521,544.20)	(509.85,462.29,**486.47**)	(584,521,559.40)	(530.64,482.31,509.31)

[a]NM: The nominal makespan value
[b]SV: The score value of the neutrosophic makespan

the crossover probabilities are set to 0.6; the scaling factor in DE and the mutation probability in GA are set to 1.0 and 0.05, respectively. As the case in the proposed TLBO algorithm, 800 individuals are employed in the three algorithms respectively and each algorithm will be stopped after 800 iterations. Five independent computations are performed for each instance. Corresponding results with comparisons are presented in Tables 4, 5, and 6. According to Table 4, the improved TLBO algorithm performs better than GA for large-scale instances, while GA performs better than the improved TLBO algorithm for some small-scale instances. This reveals that the improved TLBO algorithm has better exploration and exploitation capacities because the solution space is more complex in large-scale instances; in such a case, GA may be trapped into local optimum. Such phenomenon can also be observed in DE and PSO algorithms according to Tables 5 and 6: an individual mainly refers to or imitates the global best individual, and hence, it lacks genetic diversity. Therefore, such individuals are not likely to produce high-quality solutions. From Tables 5 and 6, it can be perceived that the improved TLBO algorithm performs totally better than the DE and the PSO algorithms.

In the following, results of medium uncertainty and small uncertainty scenarios are compared and analyzed. Based on the number of jobs in an instance, the 24 instances can be classified into three types: the small-scale, the medium-, and the large-scale instances. Typical instances covering the three types of instances are selected and the results are briefly summarized in Tables 7 and 8. Similar situations can be observed in Tables 7 and 8: the average score values of the neutrosophic makespan yielded by the improved TLBO algorithm are better than those of the plain TLBO algorithm in both medium uncertainty and small uncertainty cases. This also reflects the superiority of the improved TLBO algorithm. Also presented in the tables, the average nominal makespan values yielded by the improved TLBO algorithm are larger than the ones obtained by the plain TLBO algorithm; again, this means that a scheduling scheme with a large nominal makespan value can resist the risk of processing time fluctuations and a so-called 'optimal' scheduling scheme in the determinis-

Table 4 Results comparisons between the improved TLBO and GA with large uncertainty

Cases	The improved TLBO		GA	
	NM[a] (max, min, mean)	SV[b] (max, min, mean)	NM (max, min, mean)	SV (max, min, mean)
1	(509,465,486.20)	(422.73,373.23,392.63)	(488,446,469.60)	(369.27,321.75,**345.31**)
2	(389,362,373.40)	(323.52,308.16,**316.29**)	(379,354,368.60)	(327.69,302.94,317.07)
3	(421,360,376.40)	(348.03,326.70,341.55)	(406,371,389.20)	(346.44,321.75,**330.43**)
4	(328,319,322.80)	(306.90,269.36,295.30)	(335,314,321.40)	(292.05,251.46,**275.24**)
5	(363,330,350.40)	(318.28,277.25,**307.03**)	(370,325,344.00)	(317.52,286.16,307.14)
6	(526,470,494.20)	(387.09,343.53,**354.87**)	(538,479,501.00)	(389.07,344.52,374.56)
7	(402,376,387.40)	(373.95,315.81,333.35)	(402,383,394.00)	(348.48,307.78,**324.16**)
8	(392,357,370.80)	(339.57,315.46,324.81)	(377,363,369.80)	(321.46,310.86,**316.44**)
9	(509,466,491.80)	(340.56,297.00,**316.16**)	(509,441,475.40)	(377.29,298.96,342.33)
10	(506,452,475.20)	(430.65,389.25,415.53)	(512,480,500.00)	(438.57,370.75,**391.94**)
11	(416,366,384.00)	(365.07,342.72,352.43)	(398,365,388.80)	(349.47,344.96,**346.97**)
12	(352,333,339.40)	(327.69,313.68,322.38)	(376,333,358.40)	(326.70,303.893,**317.05**)
13	(569,482,514.80)	(428.67,360.79,**393.51**)	(538,487,508.60)	(435.60,364.32,399.39)
14	(460,387,421.80)	(378.70,337.59,362.64)	(462,400,432.80)	(365.31,327.82,**348.85**)
15	(524,478,505.40)	(362.34,345.33,**352.40**)	(500,469,490.20)	(414.81,344.54,379.04)
16	(488,462,479.80)	(437.58,398.97,422.99)	(552,477,506.80)	(433.62,387.09,**410.04**)
17	(446,401,422.80)	(380.81,350.46,**366.46**)	(444,393,429.20)	(394.02,364.80,378.28)
18	(406,364,380.20)	(346.50,324.03,**333.42**)	(398,369,382.40)	(357.39,329.31,344.62)
19	(516,471,497.00)	(446.49,396.58,427.75)	(578,483,515.60)	(453.78,389.07,**416.27**)
20	(495,423,448.00)	(400.95,370.26,**389.64**)	(464,422,449.60)	(424.20,369.27,398.87)
21	(536,485,508.60)	(404.91,378.03,**391.81**)	(508,470,490.60)	(426.69,378.69,398.52)
22	(578,524,545.20)	(439.13,409.05,**424.40**)	(552,502,530.40)	(444.76,418.84,433.42)
23	(507,437,470.20)	(422.73,401.66,**414.43**)	(489,446,468.40)	(453.42,420.31,435.23)
24	(559,521,544.20)	(509.85,462.29,**486.47**)	(591,502,541.40)	(509.02,481.89,491.20)

[a]NM: The nominal makespan value
[b]SV: The score value of the neutrosophic makespan

tic case however will result in vulnerability or fragility in the performance. This research therefore also indicates that considering only the optimal makespan value in deterministic processing time scenarios is not enough to deal with real-life situations on the shop floor. This research therefore gives some clues for the optimization method of scheduling problems with uncertain processing times.

Figure 8 gives a Gantt chart of Instance 24 with deterministic processing times (nominal processing times) and the corresponding neutrosophic version with large uncertainty is presented in Fig. 9 where the scales of x-ordinate are valued as score values of neutrosophic makespan. The neutrosophic starting times are given below the horizontal line, while the neutrosophic completion times are depicted above the line. For the purpose of a clear view, only the broken line presenting the truth membership Tx is given for each srarting/completion time in the Gantt chart. By comparing the two Gantt charts, it can be found that the processing sequences (operation permutations) on each machine in both the two Gantt charts are identical; the Gantt chart in Fig. 9 is the actual version of the one in Fig. 8. According to Fig. 8, the last operation is operation 3.19 on machine $M2$ and other operations are all completed earlier than operation 3.19; nevertheless, the Gantt chart in Fig. 9 indicates that the actual situation is different with the case in Fig. 8. Clearly, the idle time between operations on machines $2, 3, 5-8, 13-15$ is quite necessary to absorb processing time fluctuations. It can also be observed from Fig. 9 that the amplitude of the truth membership value increases continuously with the optimization process, because the truth membership values increases, while the indeterminacy and the falsity membership values decrease after each addition operation of two TNS numbers.

Conclusions

This research mainly focuses on the uncertain IPPS problem. The integration of process planning and scheduling can achieve an efficient utilization of resources to reduce conflicts in a flexible manufacturing system; nevertheless,

Table 5 Results comparisons between the improved TLBO and PSO algorithms with large uncertainty

Cases	The improved TLBO		PSO	
	NM[a] (max, min, mean)	SV[b] (max, min, mean)	NM (max, min, mean)	SV (max, min, mean)
1	(509,465,486.20)	(422.73,373.23,**392.63**)	(482,469,477.80)	(477.18,464.31,473.02)
2	(389,362,373.40)	(323.52,308.16,**316.29**)	(386,362,375.20)	(382.14,359.27,371.77)
3	(421,360,376.40)	(348.03,326.70,**341.55**)	(393,380,386.60)	(389.28,376.20,383.18)
4	(328,319,322.80)	(306.90,269.36,**295.30**)	(323,314,319.00)	(318.24,309.81,315.49)
5	(363,330,350.40)	(318.28,277.25,**307.03**)	(339,329,334.00)	(335.61,324.61,330.37)
6	(526,470,494.20)	(387.09,343.53,**354.87**)	(490,485,487.60)	(485.10,481.14,483.32)
7	(402,376,387.40)	(373.95,315.81,**333.35**)	(413,378,396.80)	(398.97,374.22,383.51)
8	(392,357,370.80)	(339.57,315.46,**324.81**)	(382,368,375.40)	(378.18,364.87,371.62)
9	(509,466,491.80)	(340.56,297.00,**316.16**)	(482,474,478.40)	(478.46,469.28,474.43)
10	(506,452,475.20)	(430.65,389.25,**415.53**)	(503,481,496.20)	(497.97,476.19,491.24)
11	(416,366,384.00)	(365.07,342.72,**352.43**)	(412,391,403.40)	(408.38,388.39,400.23)
12	(352,333,339.40)	(327.69,313.68,**322.38**)	(377,352,367.60)	(374.71,349.54,364.21)
13	(569,482,514.80)	(428.67,360.79,**393.51**)	(528,493,510.60)	(522.72,488.07,506.38)
14	(460,387,421.80)	(378.70,337.59,**362.64**)	(450,418,432.20)	(445.94,413.82,428.33)
15	(524,478,505.40)	(362.34,345.33,**352.40**)	(493,478,488.00)	(490.24,473.22,483.95)
16	(488,462,479.80)	(437.58,398.97,**422.99**)	(529,508,513.60)	(524.32,502.92,509.08)
17	(446,401,422.80)	(380.81,350.46,**366.46**)	(481,453,467.20)	(477.11,448.79,463.13)
18	(406,364,380.20)	(346.50,324.03,**333.42**)	(423,390,406.80)	(418.77,386.90,403.43)
19	(516,471,497.00)	(446.49,396.58,**427.75**)	(542,517,532.80)	(537.24,514.33,528.68)
20	(495,423,448.00)	(400.95,370.26,**389.64**)	(496,458,473.40)	(491.04,454.24,468.98)
21	(536,485,508.60)	(404.91,378.03,**391.81**)	(518,504,509.60)	(513.56,499.65,504.91)
22	(578,524,545.20)	(439.13,409.05,**424.40**)	(578,533,560.00)	(572.22,528.31,554.99)
23	(507,437,470.20)	(422.73,401.66,**414.43**)	(536,509,523.80)	(531.11,504.17,518.91)
24	(559,521,544.20)	(509.85,462.29,**486.47**)	(617,581,600.80)	(611.09,578.69,596.12)

[a]NM: The nominal makespan value
[b]SV: The score value of the neutrosophic makespan

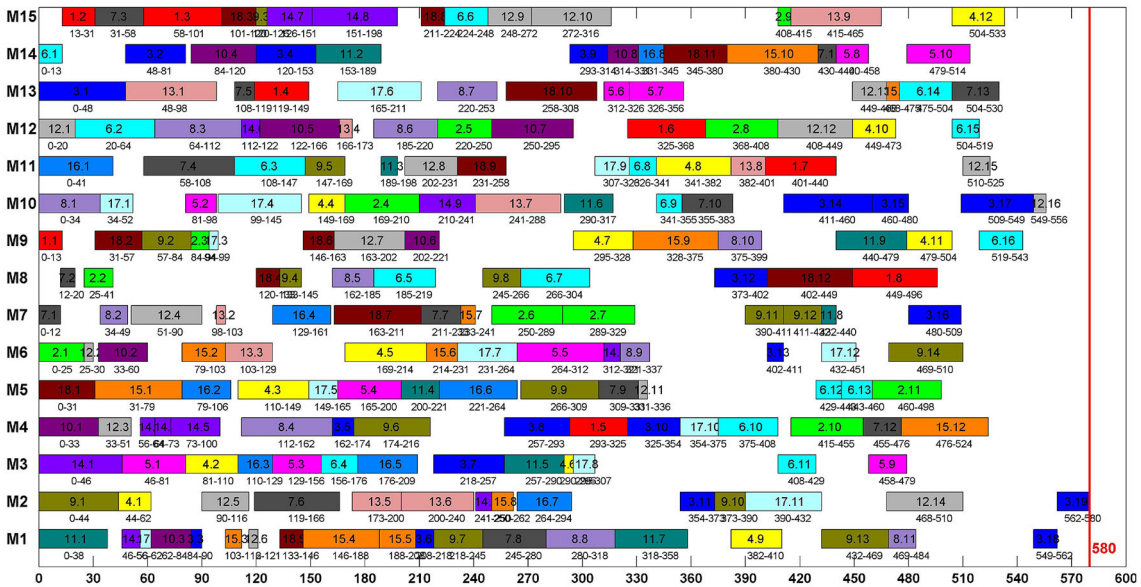

Fig. 8 The Gantt chart of Instance 24 with nominal processing times

Table 6 Results comparisons between the improved TLBO and DE algorithms with large uncertainty

Cases	The improved TLBO		DE	
	NM[a] (max, min, mean)	SV[b] (max, min, mean)	NM (max, min, mean)	SV (max, min, mean)
1	(509,465,486.20)	(422.73,373.23,**392.63**)	(611,542,592.40)	(469.26,455.40,465.30)
2	(389,362,373.40)	(323.52,308.16,**316.29**)	(590,468,533.00)	(366.30,361.35,364.06)
3	(421,360,376.40)	(348.03,326.70,**341.55**)	(648,444,545.60)	(378.18,368.28,373.82)
4	(328,319,322.80)	(306.90,269.36,**295.30**)	(556,446,512.20)	(318.86,311.79,316.32)
5	(363,330,350.40)	(318.28,277.25,**307.03**)	(513,472,500.20)	(336.34,321.75,329.27)
6	(526,470,494.20)	(387.09,343.53,**354.87**)	(746,554,666.60)	(477.18,463.32,470.05)
7	(402,376,387.40)	(373.95,315.81,**333.35**)	(656,468,554.40)	(381.15,376.20,378.23)
8	(392,357,370.80)	(339.57,315.46,**324.81**)	(576,458,524.40)	(471.24,357.39,385.80)
9	(509,466,491.80)	(340.56,297.00,**316.16**)	(792,594,639.80)	(474.21,462.89,467.85)
10	(506,452,475.20)	(430.65,389.25,**415.53**)	(848,679,760.20)	(483.75,466.29,477.70)
11	(416,366,384.00)	(365.07,342.72,**352.43**)	(755,579,677.00)	(400.05,395.01,398.09)
12	(352,333,339.40)	(327.69,313.68,**322.38**)	(599,459,549.60)	(359.15,352.44,356.51)
13	(569,482,514.80)	(428.67,360.79,**393.51**)	(865,692,773.20)	(495.09,485.10,490.21)
14	(460,387,421.80)	(378.70,337.59,**362.64**)	(747,644,681.00)	(538.41,403.92,437.38)
15	(524,478,505.40)	(362.34,345.33,**352.40**)	(749,613,682.40)	(476.19,466.29,471.83)
16	(488,462,479.80)	(437.58,398.97,**422.99**)	(973,653,819.80)	(498.26,481.31,490.82)
17	(446,401,422.80)	(380.81,350.46,**366.46**)	(845,718,784.20)	(619.74,449.46,488.66)
18	(406,364,380.20)	(346.50,324.03,**333.42**)	(693,619,648.20)	(417.01,395.35,407.95)
19	(516,471,497.00)	(446.49,396.58,**427.75**)	(891,829,855.40)	(526.30,507.87,517.74)
20	(495,423,448.00)	(400.95,370.26,**389.64**)	(782,625,717.00)	(462.33,458.15,459.92)
21	(536,485,508.60)	(404.91,378.03,**391.81**)	(872,725,796.20)	(497.97,489.27,494.65)
22	(578,524,545.20)	(439.13,409.05,**424.40**)	(998,863,915.80)	(557.37,540.60,548.96)
23	(507,437,470.20)	(422.73,401.66,**414.43**)	(1034,797,905.80)	(518.89,502.92,510.95)
24	(559,521,544.20)	(509.85,462.29,**486.47**)	(974,804,900.00)	(598.25,590.24,594.28)

[a]NM: The nominal makespan value
[b]SV: The score value of the neutrosophic makespan

Table 7 Results comparisons with medium uncertainty

Instance IDs	Total jobs	The improved TLBO		The plain TLBO	
		NM[a] (mean)	SV[b] (mean)	NM (mean)	SV (mean)
1	6	470.40	**381.40**	452.80	421.60
2	6	362.80	**312.40**	361.80	341.93
3	6	368.80	**335.20**	365.00	351.40
11	9	395.20	**350.80**	389.80	351.20
12	9	350.00	**326.07**	343.80	331.00
13	9	503.20	**400.80**	487.80	410.72
22	15	525.00	**444.46**	521.60	459.93
23	15	484.00	**430.19**	463.20	432.80
24	18	566.80	**481.15**	542.20	495.18

[a]NM: The nominal makespan value.
[b]SV: The score value of the neutrosophic makespan

due to the uncertain processing times, the implementation of the so-called 'optimal' scheduling scheme usually results in deteriorations in the the performance of the manufacturing system. To address such uncertain IPPS problems, this paper presents a TNS-based methodology. The TNSs are employed for the first time to model the uncertain processing times and a TNS-based mathematical model is also established to facilitate the problem. We further discuss the method on how to convert an MILP model of the deterministic IPPS problem into the TNS-based one. Due to the NP-hardness, the uncer-

Table 8 Results comparisons with small uncertainty

Instance IDs	Total jobs	The improved TLBO		The plain TLBO	
		NM[a] (mean)	SV[b] (mean)	NM (mean)	SV (mean)
1	6	465.80	**393.80**	462.80	407.20
2	6	373.60	**324.60**	366.00	332.60
3	6	371.80	**342.40**	357.00	346.00
11	9	377.20	**349.40**	380.60	361.80
12	9	354.20	328.80	346.60	**327.80**
13	9	496.00	**397.43**	491.40	428.60
22	15	529.00	**441.60**	529.40	477.61
23	15	469.40	**421.87**	491.80	465.72
24	18	544.40	**494.22**	527.40	498.76

[a]NM: The nominal makespan value.
[b]SV: The score value of the neutrosophic makespan

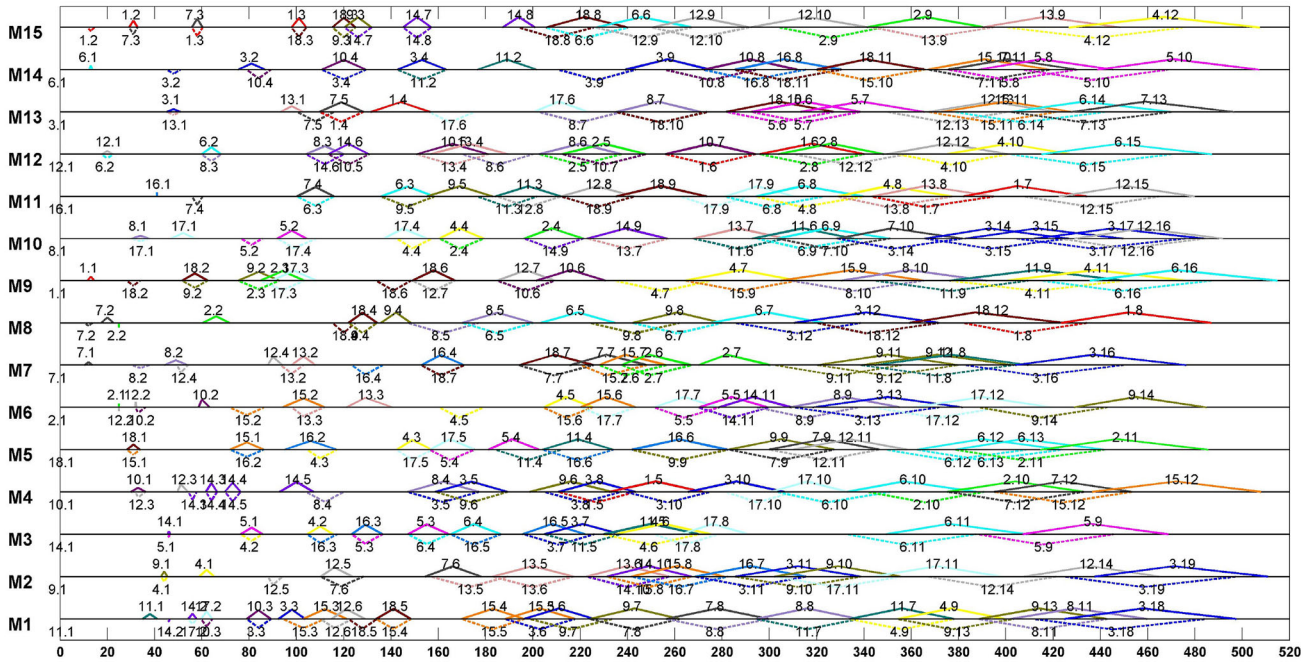

Fig. 9 The Gantt chart of Instance 24 in large uncertainty case

tain IPPS problem is solved by the improved TLBO algorithm to seek for robust solutions. The outstanding feature of the TLBO algorithm stems from parameter independency; more teachers are included in the improved TLBO algorithm to improve the individual diversity and hence intensify the search ability of the algorithm. The score function is adopted in neutrosophic makespan determination and the corresponding score value is treated as the objective. We test the Kim's benchmark in the experimental study and the results show that the improved TLBO algorithm is better than the plain TLBO algorithm in all the small uncertainty, the medium uncertainty, and the large uncertainty scenarios. More importantly, we find out that a scheduling scheme with some idle time slots may perform better, because these idle time slots can absorb processing time fluctuations. Finally, two Gantt charts with both nominal as well as neutrosophic processing times are given.

To further improve the solution quality, e.g., the quality of some small-scale instances, problem-specific neighborhood structure can be deigned and incorporated into the TLBO algorithm. In this research, the uncertain levels of neutrosophic sets in experiments are determined based on the predetermined scenarios; this relies on the actual data from shop floor in real-life applications; therefore, prior knowledge or data are required in applying neutrosophic set-based optimization method. In addition, this research considers only the neutrosophic makespan criterion. In many cases, machine utilization is another important criterion; therefore, the multi-objective uncertain IPPS problem can be listed as the future research direction. In real-life situations, there are

other kinds of uncertainties, such as machine breakdowns, random job arrivals and preventive maintenance of machines in scheduling. Such factors can be included in the uncertain IPPS problems and considered in subsequent studies.

Acknowledgements This work was supported by the Funds for the National Natural Science Foundation of China under Grants 51805330, 51575211, 51561125002, and 51905494; Zhejiang Provincial Natural Science Foundation of China under Grant No.LQ18E050006; the Youth Fund for Humanities and Social Sciences of the Ministry of Education of China under Grant 19YJCZH185; and Zhejiang Province Basic Public Welfare Research Project Program under Grant LGN19C140004.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Gao KZ, Suganthan PN, Pan QK, Tasgetiren MF (2015) An effective discrete harmony search algorithm for flexible job shop scheduling problem with fuzzy processing time. Int J Prod Res 53(19):5896–5911
2. Pezzella F, Morganti G, Ciaschetti G (2008) A genetic algorithm for the flexible job-shop scheduling problem. Comput Oper Res 35(10):3202–3212
3. Wang L, Wang S, Ye X, Zhou G, Liu M (2012) A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem. Comput Ind Eng 62(4):917–926
4. Jin L, Zhang C (2019) Process planning optimization with energy consumption reduction from a novel perspective: Mathematical modeling and a dynamic programming-like heuristic algorithm. IEEE Access 7:7381–7396
5. Chao L, Gao L, Li X, Chen P (2016) Energy-efficient multi-pass turning operation using multi-objective backtracking search algorithm. J Clean Prod 137(nov.20):1516–1531
6. Lv S, Qiao L (2013) A cross-entropy-based approach for the optimization of flexible process planning. Int J Adv Manuf Technol 68(9–12):2099–2110
7. Li X, Gao L, Wen X (2013) Application of an efficient modified particle swarm optimization algorithm for process planning. Int J Adv Manuf Technol 67(5–8):1355–1369
8. Haddadzade M, Razfar MR, Zarandi MHF (2014) Integration of process planning and job shop scheduling with stochastic processing time. Int J Adv Manuf Technol 71(1–4):241–252
9. Liu T-K, Chen Y-P, Chou J-H (2014) Solving distributed and flexible job-shop scheduling problems for a real-world fastener manufacturer. IEEE Access 2:1598–1606
10. Lian K, Zhang C, Gao L, Li X (2012) Integrated process planning and scheduling using an imperialist competitive algorithm. Int J Prod Res 50(15):4326–4343
11. Li X, Shao X, Gao L, Qian W (2010) An effective hybrid algorithm for integrated process planning and scheduling. Int J Prod Econ 126(2):289–298
12. Kumar M, Rajotia S (2006) Integration of process planning and scheduling in a job shop environment. Int J Adv Manuf Technol 28(1–2):109–116
13. Liu Q, Li X, Gao L, Li Y (2020) A modified genetic algorithm with new encoding and decoding methods for integrated process planning and scheduling problem. IEEE Trans Cybern:1–10
14. Jin L, Tang Q, Zhang C, Shao X, Tian G (2016) More milp models for integrated process planning and scheduling. Int J Prod Res 54(14):4387–4402
15. Sobeyko O, Mönch L (2017) Integrated process planning and scheduling for large-scale flexible job shops using metaheuristics. Int J Prod Res 55(2):392–409
16. Li X, Gao L, Li W (2012) Application of game theory based hybrid algorithm for multi-objective integrated process planning and scheduling. Expert Syst Appl 39(1):288–297
17. Kim YK, Park K, Ko J (2003) A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. Comput Oper Res 30(8):1151–1171
18. Jin L, Zhang C, Wen X, Christopher GG (2020) A neutrosophic number-based memetic algorithm for the integrated process planning and scheduling problem with uncertain processing times. IEEE Access 8:96628–96648
19. Joo BJ, Shim S-O, Chua TJ, Cai TX (2018) Multi-level job scheduling under processing time uncertainty. Comput Ind Eng 120:480–487
20. Pan CR, Qiao Y, NaiQi W, Zhou MC (2014) A novel algorithm for wafer sojourn time analysis of single-arm cluster tools with wafer residency time constraints and activity time variation. IEEE Trans Syst Man Cybern Syst 45(5):805–818
21. Jin L, Zhang C, Shao X, Tian G (2016) Mathematical modeling and a memetic algorithm for the integration of process planning and scheduling considering uncertain processing times. Proc Inst Mech Eng Part B J Eng Manuf 230(7):1272–1283
22. Li Z, Ierapetritou M (2008) Process scheduling under uncertainty: review and challenges. Comput Chem Eng 32(4–5):715–727
23. Jin L, Zhang C, Shao X, Yang X (2017) A study on the impact of periodic and event-driven rescheduling on a manufacturing system: an integrated process planning and scheduling case. Proc Inst Mech Eng Part B J Eng Manuf 231(3):490–504
24. Rangsaritratsamee R, Ferrell WG, Kurz MB (2004) Dynamic rescheduling that simultaneously considers efficiency and stability. Comput Ind Eng 46(1):1–15
25. Janak SL, Lin X, Floudas CA (2007) A new robust optimization approach for scheduling under uncertainty II, uncertainty with known probability distribution. Comput Chem Eng 31(3):171–195
26. Li X, Gao L, Wang W, Wang C, Wen L (2019) Particle swarm optimization hybridized with genetic algorithm for uncertain integrated process planning and scheduling with interval processing time. Comput Ind Eng 135(SEP.):1036–1046
27. Lei D (2010) A genetic algorithm for flexible job shop scheduling with fuzzy processing time. Int J Prod Res 48(10):2995–3013
28. Wang S, Wang L, Ye X, Liu M (2013) An effective estimation of distribution algorithm for the flexible job-shop scheduling problem with fuzzy processing time. Int J Prod Res 51(11–12):3778–3793
29. Smarandache F (1998) Neutrosophy: neutrosophic probability, set, and logic: analytic synthesis & synthetic analysis. American Research Press, Rehoboth, 1998

30. Smarandache F (2013) Introduction to neutrosophic measure, neutrosophic integral, and neutrosophic probability. Sitech & Education Publisher, Craiova-Columbus
31. Liang RX, Wang JQ, Zhang HY (2018) A multi-criteria decision-making method based on single-valued trapezoidal neutrosophic preference relations with complete weight information. Neural Comput Appl 30(11):3383–3398
32. Ye J (2014) Single valued neutrosophic cross-entropy for multicriteria decision making problems. Appl Math Model 38(3):1170–1175
33. Wang H, Smarandache F, Sunderraman R (2010) Single valued neutrosophic sets. Multispace Multistructure 4:410–413, 01
34. Ye Jun (2013) Multicriteria decision-making method using the correlation coefficient under single-valued neutrosophic environment. Int J Gen Syst 42(4:386–394
35. Deli I, Subas Y (2017) A ranking method of single valued neutrosophic numbers and its applications to multi-attribute decision making problems. Int J Mach Learn Cybern 8(4):1309–1322
36. Pramanik S, Mallick R (2019) Todim strategy for multi-attribute group decision making in trapezoidal neutrosophic number environment. Complex Intell Syst 5:05
37. Broumi S, Talea M, Bakali A, Smarandache F, Kishore Kumar PK(2017) Shortest path problem on single valued neutrosophic graphs. 2017 International Symposium on Networks, Computers and Communications (ISNCC), 2017, pp. 1–6. https://doi.org/10.1109/ISNCC.2017.8071993
38. Broumi S, Talea A, Smarandache F, Ali M (2016) Shortest path problem under bipolar neutrosophic setting. Conference: International Conference on Aerospace, Robotics, Manufacturing System, Mechanical Engineering, Mechatronics, Energy, Bioengineering and Neurorehabilitation ICMERA 2016
39. Said B, Deivanayagampillai N, Bakali A, Talea M, Smarandache F, Lathamaheswari M (2019) The shortest path problem in interval valued trapezoidal and triangular neutrosophic environment. Complex Intell Syst 5:391–402, 02
40. Kumar R, Edaltpanah SA, Jha S, Broumi S, Arindam D (2018) Neutrosophic shortest path problem. neutrosophic sets and systems 23: 5–15
41. Chai JS, Selvachandran G, Smarandache F, Gerogiannis VC, Son LH, Bui Q-T, Vo B (2020) New similarity measures for single-valued neutrosophic sets with applications in pattern recognition and medical diagnosis problems. Complex Intell Syst 7:703–723
42. Şahin R, Karabacak M (2020) 14—A novel similarity measure for single-valued neutrosophic sets and their applications in medical diagnosis, taxonomy, and clustering analysis. In: Smarandache F, Abdel-Basset M (eds) Optimization theory based on neutrosophic and plithogenic sets. Academic Press, pp 315–341
43. Rao RV, Savsani VJ, Vakharia DP (2011) Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43(3):303–315
44. Rao RV, Patel V (2013) Multi-objective optimization of heat exchangers using a modified teaching-learning-based optimization algorithm. Appl Math Model 37(3):1147–1162
45. Rao RV, Patel V (2013) An improved teaching-learning-based optimization algorithm for solving unconstrained optimization problems. Sci Iran 20(3):710–720
46. Tang Q, Li Z, Zhang LP, Zhang C (2017) Balancing stochastic two-sided assembly line with multiple constraints using hybrid teaching-learning-based optimization algorithm. Comput Oper Res 82(JUN.):102–113
47. Lei D, Gao L, Zheng Y (2018) A novel teaching-learning-based optimization algorithm for energy-efficient scheduling in hybrid flow shop. IEEE Trans Eng Manage 65(2):330–340
48. Shao W, Pi D, Shao Z (2018) A hybrid discrete teaching-learning based meta-heuristic for solving no-idle flow shop scheduling problem with total tardiness criterion. Comput Oper Res 94:89–105
49. Buddala R, Mahapatra SS (2019) Two-stage teaching-learning-based optimization method for flexible job-shop scheduling under machine breakdown. Int J Adv Manuf Technol 100:1419–1432
50. Zhang L, Wong TN (2015) An object-coding genetic algorithm for integrated process planning and scheduling. Eur J Oper Res 244(2):434–444
51. Jin L, Zhang C, Shao X (2015) An effective hybrid honey bee mating optimization algorithm for integrated process planning and scheduling problems. Int J Adv Manuf Technol 80(5–8):1253–1264
52. Shao X, Li X, Gao L, Zhang C (2009) Integration of process planning and scheduling—a modified genetic algorithm-based approach. Comput Oper Res 36(6):2082–2096
53. Zhang S, Wong TN (2018) Integrated process planning and scheduling: an enhanced ant colony optimization heuristic with parameter tuning. J Intell Manuf 29(3):585–601
54. Petrović M, Vuković N, Mitić M, Miljković Z (2016) Integration of process planning and scheduling using chaotic particle swarm optimization algorithm. Expert Syst Appl 64:569–588
55. Davari M, Demeulemeester E (2019) A novel branch-and-bound algorithm for the chance-constrained resource-constrained project scheduling problem. Int J Prod Res 57(4):1265–1282
56. Özcan U (2010) Balancing stochastic two-sided assembly lines: a chance-constrained, piecewise-linear, mixed integer program and a simulated annealing algorithm. Eur J Oper Res 205(1):81–97
57. Elyasi A, Salmasi N (2013) Stochastic scheduling with minimizing the number of tardy jobs using chance constrained programming. Math Comput Model 57(5–6):1154–1164
58. Liu M, Liu X, Chu F, Zheng F, Chu C (2020) Profit-oriented distributionally robust chance constrained flowshop scheduling considering credit risk. Int J Prod Res 58(8):2527–2549
59. Sakawa M, Mori T (1999) An efficient genetic algorithm for job-shop scheduling problems with fuzzy processing time and fuzzy due date. Comput Ind Eng 36(2):325–341
60. Lei D, Guo X (2012) Swarm-based neighbourhood search algorithm for fuzzy flexible job shop scheduling. Int J Prod Res 50(6):1639–1649
61. Wang L, Zhou G, Ye X, Liu M (2013) A hybrid artificial bee colony algorithm for the fuzzy flexible job-shop scheduling problem. Int J Prod Res 51(12):3593–3608
62. Gao KZ, Suganthan PN, Pan QK, Chua TJ, Chong CS, Cai TX (2016) An improved artificial bee colony algorithm for flexible job-shop scheduling problem with fuzzy processing time. Expert Syst Appl 65:52–67
63. Liu C, Zeng Q, Duan H, Zhou M, Lu F, Cheng J (2015) E-net modeling and analysis of emergency response processes constrained by resources and uncertain durations. IEEE Trans Syst Man Cybern Syst 45(1):84–96
64. Guo YW, Li WD, Mileham AR, Owen GW (2009) Applications of particle swarm optimisation in integrated process planning and scheduling. Robot Comput Integr Manuf 25(2):280–288
65. Özgüven C, Özbakır L, Yavuz Y (2010) Mathematical models for job-shop scheduling problems with routing and process plan flexibility. Appl Math Model 34(6):1539–1548
66. Tseng H-E (2006) Guided genetic algorithms for solving a larger constraint assembly problem. Int J Prod Res 44(3):601–625
67. Zhang F, Zhang YF, Nee A (1997) Using genetic algorithms in process planning for job shop machining. Evol Comput IEEE Trans 1:278–289, 12