Ding, J., Dauzère-Pérès, S., Shen, L., & Lü, Z. (2022). A Novel Evolutionary Algorithm for Energy Efficient Scheduling in Flexible Job Shops. IEEE Transactions on Evolutionary Computation. https://doi.org/10.1109/TEVC.2022.3222791

A Novel Evolutionary Algorithm for Energy Efficient Scheduling in Flexible Job Shops

Junwen Ding, Stéphane Dauzère-Pérès, Liji Shen, and Zhipeng Lü,

***Abstract*—Improving productivity at the expense of heavy energy consumption is often no longer possible in modern manufacturing industries. Through efficient scheduling technologies, however, we are able to still maintain high productivity while reducing energy costs. This paper addresses a flexible job shop scheduling problem under Time-Of-Use electricity tariffs with the objective of minimizing total energy consumption while considering a predefined makespan constraint. We propose a novel two-individual-based evolutionary (TIE) algorithm, which incorporates several distinguishing features such as a tabu search procedure, a topological order based recombination operator, a new neighborhood structure for this specific problem, and an approximate neighborhood evaluation method. Extensive experiments are conducted on widely used benchmark instances, which show that the proposed TIE outperforms traditional trajectory-based and population-based methods. We also analyze the key features of TIE to identify its critical success factors, and discuss the impact of varying key parameters of the problem to derive practical insights.**

***Index Terms*—Scheduling; Flexible job shop; Energy efficient; Optimization**

I. Introduction

MANUFACTURING enterprises nowadays face great environmental and economic challenges due to their huge energy consumption that induces both environmental impacts and significant energy costs. This pressure pushes manufacturers to consider not only production efficiency but also energy consumption in different sectors (Liu et al., 2019), as well as in various manufacturing systems (Ding et al., 2015; Zhou et al., 2019).

In the energy market, Time-of-use electricity tariff (TOU) is an adjustment method that varies the electricity prices at different times of day based on consumers' demands. The TOU pricing generally divides a day into several periods where on-peak and off-peak periods are alternatively adjacent, and assigns different prices accordingly. With a TOU scheme, customers can adjust their electricity consumption voluntarily to reduce their energy cost by shifting production from on-peak hours to off-peak hours (Schulz et al., 2020).

As a direct result of TOU tariffs, the start times of operations become variable to minimize the total energy cost. This, in turn, significantly increases the solution space. Take the flexible job shop problem (FJSP) as an example, where each operation of a job can be processed by a set of eligible machines: A solution consists in assigning operations to machines and sequencing operations on machines. The flexibility lies in the machine assignment. Considering TOU adds yet another level of flexibility to the problem. This refers to the flexible start times of operations, which is similar to a time-tabling procedure. Solving an FJSP to minimize the total energy cost now involves the determination of start times in addition to assignment and sequencing decisions.

Motivated by both academic and practical relevance, we investigate the FJSP under TOU tariffs. The objective is to minimize the total electricity cost (TEC) by optimally scheduling the jobs on the machines, such that the maximum completion time (makespan) does not exceed a given production deadline, typically the end of a scheduling period (e.g. one day or one week). Our purpose is to provide practical suggestions for decision makers to achieve a balance between productivity and the energy cost.

This approach is also known as energy-efficient scheduling which aims at a lower energy cost while providing the same level of service. In reducing energy cost, energy-efficient scheduling may also facilitate operations to be assigned to idle machines to improve resource utilization. Furthermore, machines requiring less processing time may be preferred, which then reduces the total energy consumption. In addition, energy-efficient scheduling focuses on fine-tuning production plans while requiring no further investment, which is particularly important for small and medium enterprises.

In our initial study (Shen et al., 2021), we mainly aim at refining an existing assignment and sequence for an FJSP to minimize TEC. This approach proves to achieve significant reduction in energy cost. It is, however, unable to explore larger solution spaces. In this paper, we are interested in developing metaheuristics that view TEC as the primary objective and overcome the weakness of our previous approach.

II. Literature Review

In this section, we give an overview of state-of-the-art literature on the general FJSP to minimize makespan as well as the relevant literature in the context of energy-aware scheduling (EAS).

The FJSP is a well-studied combinatorial optimization problem, which was introduced by Brucker and Schlie (1990) as an extension of the job shop scheduling problem. For the FJSP with makespan criteria, exact approaches using mixed integer programing (MIP) are proposed (Özgüven et al., 2010; Birgin

J. Ding and Z. Lü are with Huazhong University of Science and Technology, Wuhan, China, e-mail: {junwending,zhipeng.lv}@hust.edu.cn (Corresponding author: Zhipeng Lü)

S. Dauzère-Pérès is with Mines Saint-Etienne, CNRS, UMR 6158 LIMOS, Gardanne, France and Department of Accounting and Operations Management, BI Norwegian Business School, Oslo, Norway.

L. Shen is with WHU – Otto Beisheim School of Management, Vallendar, Germany.

et al., 2014; Hansmann et al., 2014). Extensive research efforts are also devoted to developing sophisticated metaheuristic approaches (González et al., 2015; Kemmoé-Tchomté et al., 2017).

Climate change challenges and soaring energy prices give compelling reasons to reduce energy consumption and carbon footprint. Different from classical job shop problem (JSP), the assignment decisions in the FJSP can impact the overall energy consumption since different machine assignments can lead to different total processing times, as well as different machine throughput rates. This thus brings great opportunities as well as challenges to energy efficient scheduling in flexible job shops.

In this area, a number of settings are considered in the literature including machine states (Wu and Sun, 2018), machine aging and maintenance (Mokhtari and Hasani, 2017), machine processing and crane transportation (Li et al., 2022), total carbon emission (Piroozfard et al., 2018), and total energy consumption threshold (Lei et al., 2019). Studies considering TOU electricity tariffs can be found in Zhang et al. (2017); Jiang and Wang (2020); Du et al. (2022); Chen et al. (2021). Moreover, similar energy considerations can also be found in flow shop scheduling problems (Schulz et al., 2020; Zhao et al., 2020, 2021a,b).

The remainder of this paper is organized as follows. In Section III, the problem is formally defined. Section IV proposes a generic tabu search algorithm with further refinements and analyses in Section V. Section VI introduces our two-individual-based evolutionary algorithm. Experimental results are reported in Section VII, which validate the effectiveness of the proposed TIE algorithm and help to derive practical insights. Section IX concludes the paper and discusses future works.

III. Problem formulation

We consider the flexible job shop scheduling problem where the total energy cost is minimized. A set J of n jobs and a set M of m machines are given. Each job i consists of a sequence (route) of n_i consecutive operations, denoted by $o \in O$, which can be processed on any machine in a subset $M(o) \subseteq M$ of compatible (also called eligible) machines.

For each operation o, let $P(o, k)$ be its processing time on machine $k \in M(o)$. In addition to the classic FJSP settings, we adopt the Time-of-Use pricing scheme TOU. Different unit electricity power costs, denoted by TOU $= \{E_1, \ldots, E_b, \ldots, E_B\}$, are present for each individual pricing period b, with b specifying the starting point of the associated time interval.

Compared to traditional scheduling problems, our primary objective is to ensure a desired throughput rate by imposing a maximum makespan $C_{max} \leq \bar{C}$. Note that it can also be viewed as a common deadline (e.g. one day) for all jobs. The planning horizon is thus given by $T = \{1, \ldots, \bar{C}\}$. In this paper, $\bar{C}$ is a relaxation of the lower bound (LB) of a problem instance, i.e., $\bar{C} = (1+\varepsilon) \cdot LB$. The focus is then on minimizing the total energy cost TEC with a constraint on the makespan. The adopted LBs are obtained by the state-of-the-art exact methods of the literature. Note that, for a considerable number of problem instances, LB is equal to the best-known upper bound. The relaxation rate ϵ is set to 0.1 according to practical experience. This setting ensures that feasible solutions are available.

We next introduce several important definitions. Let $\phi \in \Phi$ be the assignment and sequence of a resulting schedule with $C_{max} \leq \bar{C}$. Each operation o is assigned to a machine $k \in M(o)$ with a corresponding start time $s(o)$. Let $\mathcal{P}(o)$, respectively $\mathcal{S}(o)$, be the set with the direct predecessors and successors of o. By using the well-known definitions of head $r(o)$ and tail $q(o)$ in job shop scheduling, we have $s(o) = r(o)$ with

$$r(o) = \max_{o' \in \mathcal{P}(o)} \{r(o') + P(o', k')\} \tag{1}$$

$$q(o) = \max_{o' \in \mathcal{S}(o)} \{P(o', k') + q(o')\}. \tag{2}$$

Now, let us consider the case where the makespan is allowed to be extended to $\bar{C}$. The earliest start time $s^{min}(o)$ and the latest start time $s^{max}(o)$ for each operation o on the assigned machine can be expressed as follows:

$$s^{min}(o) = r(o) \tag{3}$$

$$s^{max}(o) = \bar{C} - q(o) - P(o, k) \tag{4}$$

Accordingly, We define critical operations in terms of C_{max} and TEC.

Definition III.1 (Time-critical operation). *For a given sequence with makespan C_{max}, operation o is time-critical if $s^{max}(o) - s^{min}(o) = \bar{C} - C_{max}$.*

Definition III.2 (Energy-critical operation). *For a given sequence with cost TEC, operation o is energy-critical if $s^{min}(o) < b < s^{max}(o) + P(o, k)$, where b is specified by TOU.*

Note that b here indicates the interval of TOU. For example, given that daily electricity prices are equally divided into 4 intervals (1-6, 7-12, 13-18, 19-24), then the corresponding b values are 1, 7, 13, and 19. By definition, energy-critical operations refer to operations that can cross these time points b without violating $\bar{C}$. In other words, energy-critical operations are allowed to move across different price intervals. By doing so, they can change the total energy cost.

Based on the start times of operations, we further specify the following types of schedules.

Definition III.3 (Compact schedule). *A schedule is compact if it is either left- or right-shifted.*

Definition III.4 (Partial compactness). *Given a schedule and a set $O^f \subset O$ of operations with fixed start times, the successors of operations in O^f ensure partial compactness if they are left-shifted.*

Alternatively, partial compactness can be defined on operations with fixed completion times and their predecessors.

As illustrated in Fig. 1, starting from a non-compact schedule in the left Gantt chart, all operations in the second and third intervals achieve partial compactness, whereas the start times of operations in $O^f = \{(1, 2), (2, 2), (3, 2)\}$ remain unchanged in the right Gantt chart.

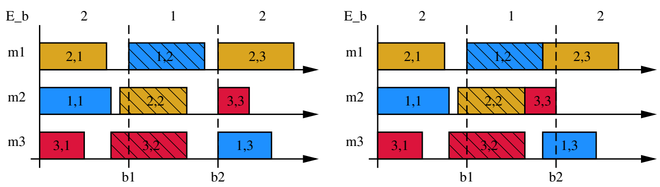


Fig. 1. Illustration of partial compactness

IV. Generic tabu search algorithm

The classical FJSP is well known to be NP-hard. We next show that our specific scheduling problem is NP-hard as well.

Proposition IV.1. *The FJSP to minimize TEC is NP-hard in the strong sense.*

Proof. Assume that we have an FJSP subject to TOU pricing where only two intervals are present. The objective is to minimize TEC. It can be reduced to the question whether it is possible to process all operations in the cheaper interval, which is a classic JSP to minimize makespan. The problem is thus NP-hard in the strong sense. □

As a first attempt to solve this NP-hard problem, we propose an initial algorithm within the basic framework of tabu search to minimize TEC while keeping $C_{max} \leq \bar{C}$. The pseudo code can be found in Algorithm 1.

Algorithm 1 The tabu search procedure

1: **Input**: Initial solution S_0, λ, $\bar{C}$
2: **Output**: The best found solution S^*
3: $S_c \leftarrow S_0$, $S^* \leftarrow S_0$, $N \leftarrow \emptyset$, $Iter \leftarrow 0$, $is\_imp \leftarrow true$
4: **while** $Iter < \lambda$ **do**
5:     **for** each time-critical operation o in S_c **do**
6:         $N \leftarrow N \cup N^k(S_c, o) \cup N^\pi(S_c, o)$
7:         $S^{'} \leftarrow \arg\min\{makespan(S) | S \in N, S \text{ is in not tabu status}\}$
8:         **if** $S^{'}$ do not exists **then**/*all neighborhood solutions in N are in tabu status*/
9:             Randomly select one solution $S^{'}$ from N
10:         **end if**
11:         Insert the move $Move(S_c, S^{'})$ into tabu list
12:         $S_c \leftarrow S^{'}$; $N \leftarrow \emptyset$
13:     **end for**
14:     **if** $makespan(S^*) > makespan(S_c)$ **then**
15:         $S^* \leftarrow S_c$; $Iter \leftarrow 0$
16:     **end if**
17:     **if** $makespan(S_c) <= \bar{C}$ **then**
18:         **while** is_imp is true **do**
19:             $is\_imp \leftarrow false$
20:             **for** each operation o in S_c **do**
21:                 $N \leftarrow N \cup N^k(S_c, o) \cup N^\pi(S_c, o)$
22:             **end for**
23:             **while** N is not empty **do**
24:                 Randomly select one solution $S^{'}$ from N
25:                 **if** $TEC(S^*) > TEC(S^{'})$ and $makespan(S^{'}) <= \bar{C}$ **then**
26:                     $S^* \leftarrow S^{'}$; $S_c \leftarrow S^{'}$; $is\_imp \leftarrow true$ **break**
27:                 **end if**
28:                 $N \leftarrow N \setminus \{S^{'}\}$
29:             **end while**
30:         **end while**
31:     **end if**
32:     $N \leftarrow \emptyset$; $Iter \leftarrow Iter + 1$
33: **end while**
34: **return** S^*

The tabu search procedure improves the solution S by re-assigning a critical operation to a different machine and inserting the operation in a feasible position, or by changing the position of a critical operation on the same machine. In this paper, the machine re-assignment is performed using the k-insertion neighborhood (called N^k), and the position change is based on the well-known N7 neighborhood proposed in (Zhang et al., 2007) (also called N^π). The tabu search procedure repeatedly chooses the best non-tabu move from $N^\pi \cup N^k$ to perform, and the move is prohibited to be reversed within the tabu tenure.

In detail, for the re-assignment move in N^k, if an operation o is removed from machine m_o, then it is forbidden to be reassigned to m_o in the next θ_1 iterations. For the re-sequencing move in N^π, we use the attribute-based tabu strategy to prevent the recent critical block to be constructed again during its tabu tenure. For example, if operation d is removed from a critical block $abcdef$ and inserted ahead of a and thus results in a new block $dabcef$, then the partial block $abcd$ is prohibited to reappear for the next θ_2 iterations.

Once the makespan of the current solution S_c is not larger than $\bar{C}$, we then optimize TEC under the constraint $makespan \leq \bar{C}$. For this purpose, we use a descend local search method with a first-improvement strategy (lines 18-30). First, we construct the neighboring solution set N from each operation of current solution S_c. Then, we randomly select one solution $S^{'}$ from N, if $TEC(S^*) > TEC(S^{'})$ and $makespan(S^{'}) \leq \bar{C}$ hold, we replace S_c with $S^{'}$. The above steps are iteratively executed until there is no improvement. The stopping condition of the tabu search procedure is the maximum number of consecutive iterations without improvement. We call this number (denoted by λ) the depth of tabu search.

V. Integrating structural properties

When solving the FJSP with TEC, not only the assignment and sequence of operations, but also their specific start times, must be considered and optimized. For better understanding, we first illustrate major differences of our current problem from the traditional job shop scheduling problem.

As a non-regular objective function, minimizing TEC does not necessarily result in left-shifted schedules. In an FJSP, the objective TEC is not equivalent to makespan minimization even for the TOU case where prices are strictly increasing or decreasing. Assume that the assignment is given for an FJSP with just two TOU intervals. Minimizing C_{max} and TEC can lead to different optimal solutions, as illustrated in Fig. 2.

Consequently, using a conventional algorithm for our scheduling problem can be time-demanding, yet less effective. To improve the algorithm performance, it is thus essential to narrow the search space. For this purpose, we consider different TOU structures and develop corresponding approaches.

A. Refining neighbourhoods N^π and N^k

Assume that an operation u is (partially) processed in interval (b_1, b_2), and it can be moved to the adjacent neighbouring intervals before b_1 and after b_2. Different pricing cases of

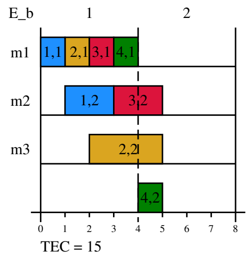


Fig. 2. Different optimal solutions for C_{max} and TEC under the same assignment

TOU are given in Fig. 3. Let $JP[u]$ and $JS[u]$ be the direct predecessor and direct successor of operation u of the same job, respectively. Similarly, $MP[u]$ and $MS[u]$ are the direct predecessor and direct successor of operation u on the same machine in a solution. If the assignment is not specifically mentioned, the simplified notation $P(u)$ is used instead of $P(u,k)$ for processing times. We can derive Lemma V.1 to suggest moving positions of u.

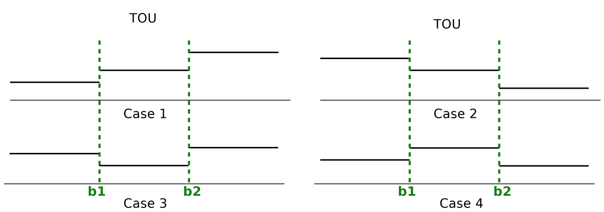


Fig. 3. Combinations of TOU with three adjacent intervals

Lemma V.1. *The energy cost of operation* u *cannot be reduced on the same machine*

1) *if* $r(MP[u]) \geq s(u)$ *holds for case 1;*
2) *if* $r(MS[u]) \leq s(u)$ *holds for case 2;*
3) *if* $b_1 \leq s(u)$ *and* $b_2 \geq s(u) + P(u)$ *hold for case 3;*
4) *if the following conditions hold for case 4*

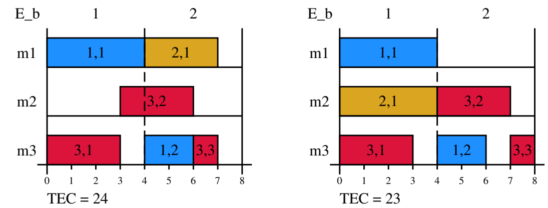
$$b_1 \leq \min\{r(JP[u])+P(JP[u]), r(MP[u])+P(MP[u])\} \quad (5)$$

$$b_2 \geq \bar{C} - \min(P(JS[u])+q(JS[u]), P(MS[u])+q(MS[u]). \quad (6)$$

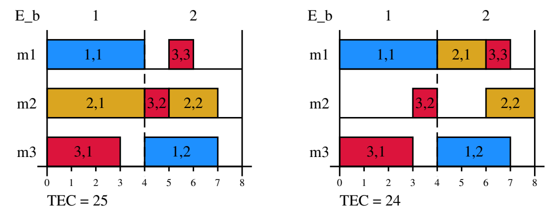
Following this result, we can refine the neighbourhood N^{π} by excluding moves that satisfy Lemma V.1. Note that this lemma addresses moves on the same machine. But moves performed on different machines (N^k) can improve TEC even if $P'(u) \geq P(u)$, where $P'(u)$ is the new processing time of u after the reassignment.

In fact, the objective TEC also has a great impact on the assignment. Occupying the cheapest interval is not always beneficial. An alternative assignment can fall into a more expensive interval, but may lead to a much smaller processing time, and thus to a smaller TEC. Fig. 4 shows two small examples to compare the optimal schedules while minimizing C_{max} and TEC, respectively. The processing time on an eligible machine may be longer but cost less. It is thus not always optimal to minimize the total processing time.

These observations suggest that the neighbourhood N^k shall be carried out more frequently. To reduce computational burden, we can now estimate TEC subject to a given assignment (ϕ).



Example 1: Assigning (2,1) to m_2 leads to longer processing time in cheaper interval. TEC is reduced by 1.



Example 2: Assigning (2,1) to m_1 leads to more expensive but shorter processing. TEC is reduced by 1.

Fig. 4. Illustration: Different optimal solutions with different assignments

Let $TP(k)$ and $TP(j)$ be respectively the total processing time on machine k and of job j for a given assignment ϕ. A lower bound on the cost for a given assignment $LB(TEC, \phi)$ can be determined as follows:

$$LB(TEC,\phi) = \max\{LB(TEC,TP(k)), LB(TEC,TP(j))\}$$
$$\forall k=1,\ldots,m, \quad j=1,\ldots,n. \quad (7)$$

where $LB(TEC,TP(k))$ and $LB(TEC,TP(j))$ are the lower bounds on TEC according to TOU while going through all machines and jobs.

More specifically, for a given assignment, the total processing times on each machine ($TP(k)$) and for each job ($TP(j)$) are determined. Assume that the processing occupies the cheapest interval of TOU first. Then, we update the remaing processing time ΔTP by subtracting the length of the cheapest interval. If there is remaining time $\Delta TP > 0$, then the processing moves to the next cheapest interval. This procedure is repeated until all operations assigned to this machine, resp. of the same job, are completed ($\Delta TP = 0$). As operation precedence of the same job or operation overlapping on machines are not considered, the resulting TEC is a lower bound of the optimal TEC.

B. New neighbourhood N^t

So far, both N^{π} and N^k focus on time-critical operations. We next propose a new neighbourhood N^t to move energy-critical operations which are essential to reduce TEC. The purpose of N^t is to adjust the start times of these operations for a given assignment and sequence ϕ. Ideally, cheaper intervals are occupied while machine idle time is pushed into expensive intervals.

Assume that we start with ϕ. For each energy-critical operation u, we determine $s^{min}(u)$ and $s^{max}(u)$. Its current start time $s(u)$ is changed by one unit each iteration until its limit is reached. After each change, TEC is evaluated and updated in the case of reduction. Depending on TOU pricing, we know that a (partially) compact schedule can be dominant. Therefore, N^t adapts to the TOU structure as depicted in Fig. 3, and enables moving blocks of operations simultaneously. After a primary move of operations u,

1) all energy-critical successors of u are pushed to the left to ensure partial compactness in case 1;
2) all energy-critical predecessors of u are pushed to the right to ensure partial compactness in case 2;
3) all energy-critical predecessors and successors of u are moved to ensure partial compactness in case 3;
4) no subsequent moves in case 4.

After the move of u, its start and completion times are fixed, which then become the latest completion time of its immediate predecessor and the earliest start time of its immediate successor. By definition of partial compactness, it is thus possible to move the corresponding predecessors/successors to ensure partial compactness, i.e. without unnecessary idle times. Especially in case 3 where the prices on both sides of adjacent intervals are higher, the purpose is to process as many operations as possible in the middle interval. Therefore, once a candidate operation u is moved, all its energy-critical predecessors and successors are moved to u as closely as possible.

C. Iterated local search

Integrating structural properties in the neighbourhood functions can, on the one hand, narrow the promising area. On the other hand, the search is more likely trapped in a smaller search space. Therefore, we adjust our previous tabu search by including a perturbation procedure while keeping the main local search structure. We denote this variant as Iterated local Search (ILS). At each iteration of ILS, Algorithm 1 is applied, followed by a perturbation procedure used in Ding et al. (2019) which randomly applies $0.2 \times |N_c|$ moves in $N^\pi \cup N^k$ on the current solution or the best found solution if the number of consecutive non-improving iterations exceeds 500, where N_c is the set of time-critical operations.

D. Approximate neighborhood evaluation of TEC

The evaluation of neighborhood solutions is one of the key components in a local search algorithm, where the quality of the neighborhood solutions is the main metric for selecting one candidate to enter the next iteration of the search process. To improve the computational efficiency, we use three methods to evaluate the TEC of the neighborhood solutions: TEC_exa, TEC_inc, and TEC_apx, whose main ideas are described below:

- TEC_exa: Exact evaluation method. After a move is applied, we calculate the actual values of r and q for all the operations, and then calculate their energy consumption.
- TEC_inc: Incremental evaluation method. In this method, we estimate the approximate values of r and q for all the impacted operations according to the topological order, and calculate the increase of their energy consumption.
- TEC_apx: Approximate evaluation method. In this method, we only estimate the approximate values of r and q for all the impacted operations on the impacted machines, and calculate the increase of their energy consumption.

Next, we elaborate on the approximate evaluation. The following method is used to estimate the value of $\hat{r}(o)$ for operation o after the move via the actual value of $r(o)$ before the move. The following three scenarios are considered:

1) *Forward insert*: Given a partial sequence $u, w_1, \ldots, w_k, v, w_s, \ldots, w_t$ on a machine, moving operation u after v results in $w_1, \ldots, w_k, v, u, w_s, \ldots, w_t$. Therefore, for operation w_1, we have

$$\hat{r}(w_1)=\begin{cases} r(JP[w_1])+P(JP[w_1]), & \text{if } u \text{ is the first operation on the machine} \\ \max\{r(JP[w_1])+P(JP[w_1]), \\ r(MP[u])+P(MP[u])\}, & \text{otherwise.} \end{cases} \tag{8}$$

For operation $o \in \{w_2, \ldots, w_k, v\}$,

$$\hat{r}(o)=\max\{r(JP[o])+P(JP[o]),\hat{r}(MP[o])+P(MP[o])\}. \tag{9}$$

For operation u,

$$\hat{r}(u)=\max\{r(JP[u])+P(JP[u]),\hat{r}(v)+P(v)\}. \tag{10}$$

For operation w_s,

$$\hat{r}(w_s)=\max\{r(JP[w_s])+P(JP[w_s]),\hat{r}(u)+P(u)\}. \tag{11}$$

For operation $o \in \{w_{s+1}, \ldots, w_t\}$,

$$\hat{r}(o)=\max\{r(JP[o])+P(JP[o]),\hat{r}(MP[o])+P(MP[o])\}. \tag{12}$$

2) *Backward insert* where operation v is moved directly before u: $\hat{r}(*)$ can be determined in a similar manner;
3) *Change machine assignment*: Assume the partial sequences $u, w_1, \ldots, w_k$ on machine m_p and $w_s, \ldots, w_t$ on machine m_q. By removing operation u from machine m_p and inserting u after w_s on machine m_q, $p \neq q$, we have

$$\hat{r}(u)=\begin{cases} r(JP[u])+P(JP[u]), & w_s \text{ is the first operation on machine } m_q \\ \max\{r(JP[u])+P(JP[u]), \\ r(MP[w_s])+P(MP[w_s])\}, & \text{otherwise.} \end{cases} \tag{13}$$

For the impacted operations $w_{s+1}, \ldots, w_t$ on machine m_q, and $w_1, \ldots, w_k$ on machine m_p, the corresponding approximate values of $\hat{r}(*)$ can also be determined in a similar manner.

The pseudo code of the proposed approximate TEC estimation method is presented in Algorithm 2, where $M_o(S)$ is the operation sequence on the assigned machine of o in solution S, and $index(o', M_o(S)$ is the position index of operation o' in the sequence $M_o(S)$. This evaluation method only calculates the change of TEC of the impacted operations on the involved machines, which improves the computational efficiency dramatically. Although this approximate evaluation method is not accurate, we do not favour the neighboring

Algorithm 2 The approximate estimation of TEC method

Input: A current solution S, and a neighboring solution $S^{'} \in N(S)$
Output: The approximate TEC value of $S^{'}$
$\Delta \leftarrow 0$ /* reset the change of TEC to 0 */
if $S^{'} \in N^{\pi}(S, o)$ **then**
  $p \leftarrow \min\{index(o, M_o(S)) | S \in \{S, S^{'}\}\}$
  for each operation $o^{'} \in \{M_o(S) | index(o^{'}, M_o(S)) >= p\}$ **do**
    estimate the approximate value $\hat{r}(o^{'})$
    $\Delta \leftarrow \Delta + EC(o^{'}, S^{'}) - EC(o^{'}, S)$
  end for
else if $S^{'} \in N^{k}(S, o)$ **then**
  $p \leftarrow \min\{index(o, M_o(S) | S \in \{S, S^{'}\}\}$
  for each operation $o^{'} \in \{M_o(S) | index(o^{'}, M_o(S)) >= p\}$ **do**
    estimate the approximate value $\hat{r}(o^{'})$
    $\Delta \leftarrow \Delta + EC(o^{'}, S^{'}) - EC(op, S)$
  end for
  for each operation
$o^{'} \in \{M_o(S^{'}) | index(o^{'}, M_o(S^{'})) >= index(o, M_o(S^{'}))\}$ **do**
    estimate the approximate value $\hat{r}(o^{'})$
    $\Delta \leftarrow \Delta + EC(o^{'}, S^{'}) - EC(o^{'}, S)$
  end for
else
  $\Delta \leftarrow EC(o, S^{'}) - EC(o, S)$ /*for N^{t} neighborhoods, only one operation is changed*/
end if
$TEC(S^{'}) \leftarrow TEC(S) + \Delta$

solution with the best improved TEC. Our preliminary test shows that a moderately improved solution appears to be more suitable for speeding up the search.

VI. Two-Individual-based Evolutionary Algorithm

A. Motivation

As one of the most difficult combinatorial problems, the classical FJSP is hard to solve, and TOU adds new challenges to the design of solution approaches. As discussed in Section I, solving the FJSP with TOU involves three decisions: Machine assignment, operation sequencing, and operation time-tabling. Especially the newly incurred time-tabling subproblem further expands the solution space. Therefore, computational times become a critical issue. After refining and extending our tabu search, we integrate further elements to strengthen our algorithm. It is denoted by Two-Individual-based Evolutionary algorithm (TIE).

Being a trajectory-based search technique, tabu search follows the inner properties of the solution, and neglects the implicit relationships among the solutions. Traditional population-based algorithms may compensate this. They, however, have to deal with a large population accompanied by higher computational times. The drawbacks of each type of algorithms motivated us to pursue an alternative combination of trajectory and population-based methods.

More specifically, we focus on two individuals, which is a unique feature of TIE. These two individuals simulate the behaviour of people with their predecessors, and are respectively responsible for intensification and diversification. They evolve for a given number of generations, known as a cycle. At the end of each cycle, one individual is replaced by the predecessor in the previous cycle to continue the process to absorb the essence of the evolution history. The other individual goes through a specific procedure to ensure diversity. In the following, we present the general architecture as well as different components of TIE.

B. Basic framework

Based on two individuals, TIE follows the basic framework of the evolutionary algorithms (Lü et al., 2010; Sutton and Neumann, 2012; Zhao et al., 2020). Its diagram is depicted in Fig. 5 and its general architecture can be found in Algorithm 3. The algorithm consists of three main components: The *Init()* function to generate a random solution, the tabu search procedure *TS(S)* to improve the solution S, and the topological order based recombination operator *TOCX* to generate two child solutions. A cycle consists of generations of length p, where p is an integer parameter. The best solution in the current (previous) cycle is stored in $S_c^* \ (S_p^*)$.

First, TIE uses the *Init()* procedure to generate random solutions by assigning each operation of each job to its candidate machines with equal probability subject to all the constraints. As a result, initial solutions are obtained for S_1, S_2, S_c^*, S_p^* and S^*, where S_c^*, S_p^* and S^* are to be replaced subsequently.

Next, at each generation, TIE adopts *TOCX* on S_1 and S_2 to generate two child solutions $S_1^{'}$ and $S_2^{'}$, which are then optimized by the tabu search procedure to become new solutions S_1 and S_2.

At the end of each cycle, we update S_1 by the best solution S_p^* found in the previous cycle, while S_c^* is initialized as a random solution before entering the next cycle. Moreover, S_2 is replaced with a random solution once it becomes too close to S_1 and impairs the search diversity.

In this context, two solutions S_1 and S_2 are considered to be close when the number of operations $d(S_1, S_2)$ that have different machine assignments or different positions on the same machine is smaller than a threshold value δ of the total number of operations of all jobs. TIE terminates when its running time exceeds a predefined value T_{max} seconds.

In detail, the solution representation of FJSP in TIE takes the form $S = (\phi, \pi)$, where ϕ is a feasible assignment and π the processing order on machines. More specifically, $\phi(o, S) = k$ indicates that operation o is assigned to machine $k \in M(o)$ according to S whereas $\pi(o, S) = i$ means that operation o is sequenced in the i-th position of its assigned machine k. Let $\beta(o)$ be a binary variable that indicates whether or not operation o is on the same machine and same position of two solutions S_1 and S_2, which are formally defined as follows:

$$\beta(o) = \begin{cases} 1 & \text{if } \phi(o, S_1) = \phi(o, S_2), \pi(o, S_1) = \pi(o, S_2), \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

Therefore, the number of operations that have different machine assignments or different positions on the same machine can be expressed as $d(S_1, S_2) = \sum_{o \in O} \phi(o)$. If $d(S_1, S_2) < \delta \cdot \sum_{i=1}^{n} n_i$ holds, the two solutions S_1 and S_2 are deemed close to each other.

C. Topological order based recombination operator

Algorithm 4 presents our recombination operator based on a topological order. The operator uses a parameter γ, whose

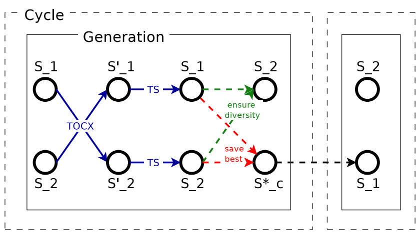


Fig. 5. General framework of TIE

Algorithm 3 TIE for FJSP with TOU scheme

Input: Problem instance
Output: The best solution S^* found
$gen \leftarrow 0$; $S_1, S_2, S_c^*, S_p^*, S^* \leftarrow$ Init()
while stopping condition is not reached **do**
  $S_1^{'} \leftarrow$ TOCX(S_1, S_2), $S_2^{'} \leftarrow$ TOCX(S_2, S_1)
  $S_1 \leftarrow$ TS($S_1^{'}$), $S_2 \leftarrow$ TS($S_2^{'}$)
  $S_c^* \leftarrow$ save_best(S_1, S_2, S_c^*)
  $S^* \leftarrow$ save_best(S_c^*, S^*)
  if gen is equal to an integer parameter p **then**
    $S_1 \leftarrow S_p^*$, $S_p^* \leftarrow S_c^*$, $S_c^* \leftarrow$ Init(), $gen \leftarrow 0$
  end if
  if $S_1 \approx S_2$ **then**
    $S_2 \leftarrow$ Init()
  end if
  $gen \leftarrow gen + 1$
end while
return S^*

Algorithm 4 Topological order recombination (TOCX)

1: **Input**: parent solutions S_1 and S_2, γ
2: **Output**: An offspring solution S_o
3: calculate the topological order T_1 of S_1
4: calculate the topological order T_2 of S_2
5: an empty topological order list T, $\varphi \leftarrow 0$
6: **while** $\varphi <=$ total number of operations **do**
7:   choose the first operation o from $T_{\varphi/2+1}$
8:   $N \leftarrow \emptyset$
9:   **for** each position i in list T **do**
10:     insert o into position i of T, results in a sub list T_o^i
11:     record the corresponding machine assignment $\phi(o, S_{\varphi/2+1})$ for o in list T_o^i
12:     mapping T_o^i into a sub solution S_o^{i}
13:     **if** S_o^{i} is feasible **then**
14:       $N \leftarrow N \cup \{S_o^i\}$
15:     **end if**
16:   **end for**
17:   **if** $rand(0,1) < \gamma$ **then**
18:     $S_o^{imin} \leftarrow \arg\min\{makespan(S_o^i) | S_o^i \in N\}$
19:   **else**
20:     randomly select a solution S_o^{imin} from N
21:   **end if**
22:   insert o into position i_{min} of list T
23:   record the corresponding machine assignment $\phi(o, S_{\varphi/2+1})$ for o in list T
24:   remove o from T_1 and T_2
25:   $\varphi \leftarrow \varphi + 1$
26: **end while**
27: mapping T into a complete solution S_o
28: **return** S_o

value is established empirically later. The offspring solution S_o is built step by step from two parent solutions S_1 and S_2 as follows. First, we calculate the topological order T_1 and $T2$ for S_1 and S_2, respectively. Second, we choose one operation o alternatively from T_1 and T_2, insert o into each position i of list T, and record the corresponding machine assignment, which corresponds to a partial solution S_o^i. S_o^i is collected in set N if it is feasible. Next, we generate a random number in the range of $[0, 1]$. If this number is smaller than γ, we insert o into the position of T having the smallest objective value. Otherwise, operation o is inserted into a random position which leads to a feasible solution. The above steps are repeated until all operations have been considered. Finally, T with the machine assignment information is mapped into a complete solution S_o.

The inner *for* loop iterates at most $|T|$ times (line 9), the makespan of a partial solution can be calculated in linear time (line 18), and the outer *while* loop iterates n_o times, therefore, the time complexity of TOCX is $O(n_o^2)$, where n_o is the total number of operations, i.e., $n_o = \sum_{i=1}^{n} n_i$.

TABLE I

PARAMETER SETTINGS IN TIE

Para.	Section	Description	Value
ε	III	relaxation rate	0.1
η	VII-A	convert rate	0.1, 0.01[a]
p	VI-B	cycle length	10
δ	VI-B	similarity threshold	0.1
γ	VI-C	recombination rate	0.3
θ_1	IV	tabu tenure for N^k	$m+rand()\%(2*m)$
θ_2	IV	tabu tenure for N^π	$n+rand()\%n$
λ	IV	depth of tabu search	10000
T_{max}	VI	maximum run time of TIE	600

[a] For *DPdata* and *DMUdata*, we set $\eta = 0.01$ since the makespan is relatively larger, and for the other instances, we set $\eta = 0.1$.

VII. Experiment Design

A. Parameter settings and experimental protocol

In subsequent sections, we report extensive numerical results to tackle 8 sets of a total of 393 benchmark FJSP instances widely used in the literature. We coded TIE algorithm in C++ and ran it on a cluster of Intel Xeon E5-2697 processor with 2.60 GHz CPU and 2 GB RAM. Table I gives the descriptions and settings of the parameters used in TIE, where the last column denotes the settings for the set of all the instances. Given the stochastic nature of TIE, we solved each problem instance ten times independently.

To simulate the Time-of-Use electricity tariffs in real world industries, we adopt eight different TOUs in our experiments as given in Table II. Each TOU has 4 basic intervals, and each interval represents 6 hours. Therefore, a complete TOU cycle has 24 hours. For example, in TOU0, the electricity prices in 1-6 hours, 7-12 hours, 13-18 hours, and 19-24 hours are 1, 2, 1, and 4 units, respectively.

This pricing setting simulates the energy cost in real life, where on-, mid-, and off-peak prices are present. If the planning horizon, i.e. makespan, exceeds 24 hours, the TOU setting repeats according to the same scheme. Note that scheduling activities usually have a short planning horizon of one day or several days. On the other hand, the makespan values of the benchmark sets vary greatly. For example, the makespan of instances in *DPdata* ranges from 2000 to 2500, while the majority of the instances in the other sets have a makespan smaller than 1000. Therefore, we adopt a parameter convert

TABLE II
DIFFERENT TOU SETTINGS IN TIE

TOU	Value	TOU	Value
TOU0	1, 2, 1, 4	TOU4	4, 1, 2, 1
TOU1	1, 2, 3, 2	TOU5	2, 3, 2, 1
TOU2	1, 2, 3, 4	TOU6	4, 3, 2, 1
TOU3	2, 1, 3, 4	TOU7	4, 3, 1, 2

rate η to transform the makespan to real time in hours: $\eta = 0.01$ for *DPdata* and *DMUdata*, and $\eta = 0.1$ for the other instances.

B. Benchmark instances

In this study, we employed five sets of benchmark instances to assess the performance of our TIE algorithm: *DPdata* (Dauzère-Pérès and Paulli, 1997), *BCdata* (Barnes and Chambers, 1998), *BRdata* (Brandimarte, 1993), *HUdata* (Hurink et al., 1994), and *DMU* (Demirkol et al., 1998), having 393 instances in total with different sizes and flexibility levels, where the first four sets of benchmark instances are available in Monaldo Mastrolilli's web page[1], and the last set of benchmark instances is available in Oleg Shylo's web page[2]. The detailed information of the above benchmarks is presented in Table III, where column *flexibility* shows the average number of candidate processing machines for the operations.

C. Reference algorithms

- **Efficient heuristics (SP, DP).** The heuristics refer to the Shifting Procedure SP and Dynamic-programming-based Procedure DP (Shen et al., 2021). Both heuristics operate on the existing assignment and sequence of an FJSP. While SP systematically shifts the start times of operations to reduce TEC, DP successively builds and assesses partial solutions until a complete solution is reached with the smallest TEC.
- **Hybrid Tabu Search (TS(SP), TS(DP)).** This approach hybridizes the tabu search of Shen et al. (2018) and the SP, respectively DP. Once the tabu search finds a local optimum, we activate SP/DP to refine and improve the current best known solution, which is then returned to tabu search for the next iteration.
- **Adjustment Strategy (AS) of Jiang and Wang (2020).** The hybrid multi-objective evolutionary algorithm based on decomposition (HMOEA/D) proposed in Jiang and Wang (2020) is a multi-objective algorithm evaluated with different performance metrics. Although the procedures and problem settings are not directly comparable to ours, we adopt the crucial parts regarding TOU in HMOEA/D, i.e., the adjustment strategy for comparison. Since HMOEA/D is not applicable to our problem, and also to ensure a fair comparison, the adjustment strategy starts with the solution generated by the tabu search of Algorithm 1, which is of good quality and satisfies $C_{max} \leq \bar{C}$. The adjustment strategy itself consists of three main steps:
  1) For a given solution, find critical paths;
  2) Determine the machine k_{max} with the highest electricity cost;
  3) Starting from the last operation on k_{max}, if it is not time-critical, then shift this operation to a possible lowest price interval. This procedure stops once all operations on k_{max} are considered.
- **ILS** as described in Section V-C.
- **TIE** as described in Section VI.
- **Hybrid Evolutionary Algorithm (HEA)** follows the framework of Ding et al. (2017). The purpose of HEA is to compare with the two-individual structure. Therefore, it builds a large population for evolution while keeping the other elements similar to TIE.
  During each generation, HEA employs TOCX to obtain a new offspring solution from randomly selected individuals with a probability of 0.6. Otherwise, it uses a perturbation procedure to mutate one individual into a new offspring solution. Subsequently, local search is applied to improve the newly generated solution. As for population updating strategy, it replaces inferior individuals with better ones. For non-improving offspring, they are selected at a small probability of 0.3.

[1]https://people.idsia.ch// monaldo/fjsp.html
[2]http://optimizizer.com/DMU.php

VIII. COMPUTATIONAL RESULTS

A. Computational efficiency of TIE

To analyze its performance, we first apply TIE to the classical FJSP instances to minimize the makespan as the baseline criterion for future comparison. Note that the tabu search procedure in Algorithm 1 can be viewed as a two-phase approach. In the first part, it optimizes the makespan until S_c is not larger than $\bar{C}$ (lines 5-16). This is to ensure that the prerequisite of our FJSP setting is satisfied. Afterward, it minimizes TEC under the constraint $makespan \leq \bar{C}$ (lines 17-31). Therefore, the first phase is equivalent to an algorithm for makespan minimization. According to our experiments, it reaches similar best-known solutions as the MAE in Ding et al. (2019). It matches best-known upper bounds for most benchmark instances.

To analyze the impact of the computational time on the performance of TIE, we test TIE on all the benchmark instances with TOU0 under five different cutoff times: 1, 5, 10, 20, and 30 minutes, and each instance is run for 10 independent times. Afterwards, we apply Wilcoxon's signed rank tests (Wilcoxon, 1992) on each pair of cutoff times for multiple comparisons. The corresponding statistical results of p-values are reported in Table IV. It shows that the performance of TIE is significantly different with computational times lower than 10 minutes. The additional improvement obtained by TIE is not significant at a confidence level of 5% when larger computational time (more than 10 minutes) is used.

Based on these results, we set the maximum computational time for TIE as 10 minutes in our main experiments.

TABLE III
DESCRIPTIONS OF THE BENCHMARK SETS

set	size	jobs n	machines m	operations n_i	Processing time	flexibility
DPdata	18	{10, 15, 20}	{5, 8, 10}	[15, 25]	[10,100]	[1.13, 5.02]
BCdata	10	{10, 15, 20}	[4, 15]	[5, 15]	[1, 20]	[1.43, 4.1]
BRdata	21	{10, 15}	[11, 18]	[10, 15]	[5, 100]	[1.07, 1.3]
HUdata/sdata	66	[6, 30]	[4, 15]	[4, 15]	[10, 100]	{1}
HUdata/edata	66	[6, 30]	[4, 15]	[4, 15]	[10, 100]	[1, 1.15]
HUdata/rdata	66	[6, 30]	[4, 15]	[4, 15]	[10, 100]	[1, 2]
HUdata/vdata	66	[6, 30]	[4, 15]	[4, 15]	[10, 100]	[1, 7.5]

TABLE IV
THE p-VALUES OF WILCOXON'S SIGNED RANK TESTS ON THE CUTOFF TIMES OF TIE

	1 min.	5 min.	10 min.	20 min.	30 min.
1 min.	-	0.0002	0.0002	0.0002	0.0002
5 min.	-	-	0.0004	0.0002	0.0003
10 min.	-	-	-	0.0512	0.0576
20 min.	-	-	-	-	0.0581
30 min.	-	-	-	-	-

B. Comparison with reference algorithms

We next compare the performance of TIE with the reference algorithms listed in Section VII-C. Table V reports the results (RPD %) of SP, DP and their hybrids TS(DP) and TS(SP), as well as TIE, where the results are sorted by benchmark sets and TOU-settings. The RPD is determined as follows:

$$RPD = 100 \cdot \left(TEC^A / TEC_{min} - 1\right) \qquad (15)$$
$$A \in \{SP; DP; TS(SP); TS(DP); TIE\},$$

where TEC^A and TEC_{min} are the TEC value of the corresponding algorithm and the minimum TEC of all considered methods.

Since TS(SP) outperforms the other heuristics, let us focus the analysis on the comparison of TS(SP) and TIE. As discussed earlier, TIE achieves major improvements within 10 minutes. We thus set CPU times to 1 and 10 minutes for both algorithms. Note that SP and DP operate on an existing solution of the FJSP instance and take 40 seconds on average.

In general, TS(SP) reaches better solutions than TIE within one minute. This is probably due to the embedded efficient heuristic SP. However, this advantage fades with more computational time since, in 10 minutes, TS(SP) provides slight improvements for a majority of instances. In comparison, TIE achieves substantial improvements with 10 minutes, and has the smallest RPD on average. This also suggests that population-based components are effective for solving this FJSP with huge solution space.

It is also worth mentioning that, when examining the benchmark sets individually, TS(SP) outperforms TIE for *BRdata, DPdata*, and *HUdata/vdata*. We assume it is due to the settings on processing time and resource flexibility of these instances. *BRdata* and *DPdata* both have non-identical processing times on eligible machines which also have relatively large range, i.e. [5,100] and [10,100]. A new assignment thus has different processing times, which likely leads to an increased number and duration of idle times on machines. The heuristic SP is able to step-wise shift movable operations on different machines simultaneously and to examine a large number of combinations of operation start times. The N^t in TIE, however, only moves a block of related operations, which may miss desirable combinations.

As for *HUdata*, processing times remain identical on all eligible machines. With moderate resource flexibility, TIE performs better. However, *HUdata/vdata* has a high flexibility of [1,7.5]. A new assignment can again provide a large number of potential combinations of start times. As discussed, this is ideal for TS(SP). It is thus our conjecture that the performance of TS(SP) and TIE is related to both processing time and resource flexibility. When both are increased to an extent, the instances become particularly challenging for TIE. This may also explain the good performance of TIE on *BCdata* which have a relatively high flexibility ([1.43,4.1]) but a small processing range ([1,20]).

When sorted by TOUs, TIE performs consistently well except for TOU7 of (4,3,1,2). Starting with a left-shifted schedule, the decreasing pricing with an embedded case 3 as in Fig. 3 is difficult for TIE to solve compared to TS(SP). The latter can move a large number of operations simultaneously to the right while step-wise movement avoids pushing many operations into the expensive (last) interval.

These results suggest that while the metaheuristic framework and neighbourhood structure are powerful and well suited for hard problems, it would also be worthwhile to investigate a combination of efficient heuristics.

Finally, we compare TIE with the adjustment strategy embedded in HMOEA/D of Jiang and Wang (2020). Note that a small portion of the benchmark instances are tested in Jiang and Wang (2020) subject to one TOU setting. We therefore use the same instances and a similar TOU structure in this comparison. Table VI reports the results of TIE and AS with TOU0, where AS is applied to the solution generated by the tabu search of Algorithm 1. We observe from Table VI that TIE outperforms AS for all the tested instances with smaller TEC_{min} and TEC_{avg} values. There is a remarkable difference between TIE and AS on instances *orb1-orb10*, where the resource flexibility is relatively smaller than that in the other instances. This indicates the robustness of TIE which can identify promising solutions for instances with a wide range of features.

C. Effectiveness of two-individual-based framework

As shown in the previous section, TIE reaches good solutions compared with state-of-the-art reference algorithms. In

TABLE V
COMPARISON (RPD %) WITH SP, DP, AND THEIR HYBRIDS

Ins.	SP	DP	TS(DP)	TS(SP)		TIE	
				1 m.	10 m.	1 m.	10 m.
BCdata	3.07	3.61	3.35	2.20	2.10	0.93	**0.04**
BRdata	15.44	15.25	3.02	2.55	**1.12**	3.31	1.65
DPdata	0.67	0.36	0.69	0.61	**0.51**	2.24	0.92
HUdata/sdata	3.21	2.95	2.27	1.57	1.39	1.21	**0.37**
HUdata/edata	2.79	2.31	2.22	1.63	1.51	1.00	**0.29**
HUdata/rdata	2.09	1.63	1.40	1.00	0.82	1.49	**0.78**
HUdata/vdata	2.77	2.32	1.03	0.77	**0.37**	1.91	1.18
Mean	3.03	2.69	1.82	1.31	1.07	1.48	**0.66**

TOU	SP	DP	TS(DP)	TS(SP)		TIE	
				1 m.	10 m.	1 m.	10 m.
TOU0	6.27	3.82	2.60	1.94	1.66	1.81	**0.67**
TOU1	2.38	2.34	1.40	0.97	0.77	1.02	**0.40**
TOU2	2.49	1.45	1.51	1.20	0.93	1.64	**0.86**
TOU3	4.05	3.31	1.72	1.15	0.90	1.17	**0.44**
TOU4	2.53	2.63	1.30	1.93	1.60	2.46	**1.23**
TOU5	1.94	2.25	2.70	0.84	0.66	0.97	**0.39**
TOU6	2.70	3.22	1.63	1.47	1.24	1.21	**0.48**
TOU7	1.84	2.52	1.71	0.99	**0.78**	1.57	0.83
Mean	3.03	2.69	1.82	1.31	1.07	1.48	**0.66**

TABLE VI
COMPARISON BETWEEN TIE AND TIE_AS WITH TOU0

Ins.	AS		TIE	
	TEC_{min}	TEC_{avg}	TEC_{min}	TEC_{avg}
Mk01	16.20	16.27	15.40	15.49
Mk02	14.60	14.67	14.40	14.48
Mk03	128.60	131.37	117.60	119.41
Mk04	37.30	37.34	36.20	36.39
Mk05	88.50	88.77	88.00	88.17
Mk06	42.20	42.51	40.80	41.13
Mk07	92.50	94.68	89.20	90.71
Mk08	407.00	417.79	368.30	371.12
Mk09	397.40	403.54	367.00	372.99
Mk10	307.20	308.58	303.40	306.14
01a	182.59	184.10	176.34	177.35
02a	199.12	200.23	197.98	197.99
03a	198.84	199.21	197.98	197.98
04a	182.23	183.36	174.95	176.20
05a	200.72	201.07	199.37	197.73
06a	198.08	198.77	198.43	198.56
07a	299.09	302.21	294.60	289.31
08a	275.56	275.79	275.76	275.94
09a	275.42	275.61	275.40	275.62
10a	302.95	304.11	297.82	291.87
orb1	972.60	977.74	910.10	912.74
orb2	921.30	925.63	853.90	858.26
orb3	958.00	962.70	865	869.95
orb4	1020.00	1026.92	938.20	941.26
orb5	858.70	861.86	798	800.44
orb6	1026.60	1030.88	938.30	951.1
orb7	384.70	387.67	356.50	360.37
orb8	806.80	815.80	729.80	735.91
orb9	940.50	948.30	851.10	855.43
orb10	1010.80	1017.27	916.20	920.79
Avg.	385.53	388.19	378.27	380.00

the following, we closely examine the performance of the two-individual structure.

First, we compare TIE with ILS, a conventional trajectory method discussed in Section VII-C. Note that the latter adopts the same key components including neighbourhood functions as TIE, which allows us to separate the effect of the two-individual structure. Our tests are conducted on all benchmark sets where consistent results are observed. For illustration, Table VII presents the results for *DPdata* with TOU0, where each instance is solved 10 independent times with a cutoff time of 10 minutes. Columns TEC_{sd}, MK_{min} and MK_{avg} denote the standard deviation of TEC, minimum makespan, and mean makespan, respectively. TIE outperforms ILS in terms of both makespan and TEC. This suggests that TIE, which focuses on two solutions instead of a single one, is superior to the trajectory method.

We next compare TIE with a population-based metaheuristic. For this purpose, we utilize the HEA as described in Section VII-C with a population of 20 solutions. Other key components remain similar for TIE and HEA, which allows us to validate the effectiveness of the two-individual-based framework

Both algorithms are tested on all benchmark instances with all TOU-settings, and show consistent results. A comparison of TIE and HEA with TOU0 on the *DPdata* instances can also be found in Table VII. It can be seen that TIE obtains better results on both TEC and makespan, which again implies that using two individuals suffices compared to a larger population.

Table VIII summarizes all results including the mean CPU time required by each method to find the best solution ($time_{avg}$), as well as PRD. PRD is calculated according to Equation (15) with $A \in \{ILS; HEA; TIE\}$. Overall, TIE has the smallest deviation while using slightly more time. The examination of detailed solutions confirms that ILS and HEA converge quickly but are unable to escape local optima afterwards.

D. Function of algorithm components

*1) **Different initial procedures**:* To explore how different initial procedures impact the performance of TIE, we implement two initial solution procedures: *total_random* and *greedy_random*. Procedure *total_random* generates random solutions by assigning each operation to one of its candidate machines with equal probability. Procedure *greedy_random* generates solutions by assigning 90% of the operations of each job to their best candidate machine, which results in the minimum TEC for the partial solutions, and 10% of the operations of each job to one of their candidate machines with equal probability.

The results of TIE and ILS with the two initial solution procedures on instances 01a and 09a are plotted in Fig. 6a and Fig. 6b, respectively. It can be seen that, the better initial solutions generated by Procedure *greedy_random* help to reach better solutions for ILS and TIE faster than with Procedure *total_random*. However, TIE with different initial procedures

TABLE VII
COMPARISON OF TIE, HEA, AND ILS WITH TOU0 ON *DPdata*

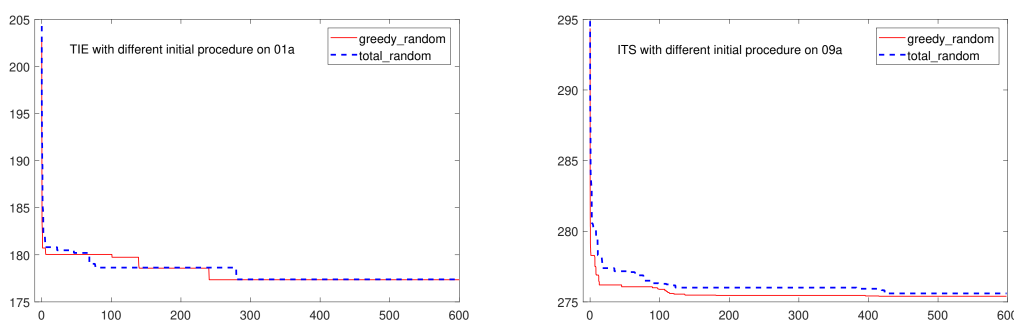
Ins.	ILS					HEA					TIE				
	TEC_{min}	TEC_{avg}	TEC_{sd}	MK_{min}	MK_{avg}	TEC_{min}	TEC_{avg}	TEC_{sd}	MK_{min}	MK_{avg}	TEC_{min}	TEC_{avg}	TEC_{sd}	MK_{min}	MK_{avg}
01a	176.31	177.99	0.87	2755	2755	176.14	177.43	0.8	2755	2755	176.29	177.35	0.67	2546	2653.8
02a	197.98	198.07	0.07	2450	2450	197.98	198.18	0.33	2450	2450	197.98	197.99	0.01	2450	2450
03a	197.98	197.98	0.01	2450	2450	197.98	198	0.04	2450	2450	197.98	197.98	0	2450	2450
04a	175.68	176.54	0.65	2753	2753	175.29	176.75	0.86	2753	2753	174.77	176.2	0.98	2565	2646.9
05a	199.17	199.85	0.34	2411	2411	199.63	200.06	0.27	2411	2411	197.1	197.73	0.37	2314	2388.3
06a	198.2	198.96	0.33	2379	2379	198.74	199.31	0.37	2204	2363.09	197.92	198.56	0.37	2256	2319.8
07a	294.58	297.91	1.5	2437	2437	295.23	299.41	1.83	2437	2437	286.2	289.31	1.77	2337	2404
08a	275.7	276.07	0.25	2074	2164.27	275.91	276.26	0.2	2081	2155.27	275.63	275.94	0.17	2072	2167.6
09a	275.61	275.8	0.16	2072	2128.36	275.61	275.77	0.12	2087	2190.36	275.46	275.62	0.07	2068	2121.1
10a	296.24	299.9	1.96	2433	2433	296.19	300.14	2.57	2433	2433	289.25	291.87	1.51	2347	2406.3
11a	270.87	272.71	0.71	2219	2219	272.48	273.55	0.72	2219	2219	271.35	272.57	0.82	2073	2187
12a	263.33	265.11	1.03	2165	2165	265.13	266.13	0.76	2165	2165	263.72	264.53	0.62	2135	2167.4
13a	395.17	397.15	1.65	2416	2416	394.81	396.34	0.62	2416	2416	381.22	382.83	0.82	2301	2416.2
14a	384.61	385.03	0.25	2173	2255.27	384.81	385	0.14	2187	2272.27	384.56	384.85	0.2	2169	2226.4
15a	384.58	384.83	0.18	2174	2254.55	384.63	384.88	0.2	2179	2249.55	384.68	385.12	0.27	2173	2314.9
16a	391.44	395.35	2.5	2412	2412	394.73	397.02	1.06	2412	2412	380.18	383.12	1.33	2316	2392.9
17a	373.5	374.89	0.8	2154	2283.09	374.5	375.74	0.91	2296	2296	373.94	375.19	1.11	2161	2289.8
18a	371.64	373.23	0.85	2262	2262	371.58	373.61	1.52	2262	2262	371.24	372.79	1.17	2140	2230.1
Avg.	284.59	285.97	0.78	2343.83	2368.2	285.08	286.31	0.74	2344.28	2371.64	**282.19**	**283.31**	**0.68**	**2270.72**	**2346.25**

Friedman's test is conducted on TEC_{avg} obtained by TIE, HEA, and ILS on *BCdata*, *BRdata*, and *DPdata*.
The resulting small value of $p = 8.8827e-06 << 0.001$ indicates that the three optimization frameworks distinguish each other statistically.

TABLE VIII
SUMMARY OF TEST RESULTS OF TIE, HEA, AND ILS

TOU	ILS				HEA				TIE			
	TEC_{avg}	TEC_{sd}	$time_{avg}$	RPD	TEC_{avg}	TEC_{sd}	$time_{avg}$	RPD	TEC_{avg}	TEC_{sd}	$time_{avg}$	RPD
TOU0	724.69	4.03	298.35	0.99	724.29	3.69	311.67	0.93	717.59	3.20	318.78	0.00
TOU1	771.54	2.53	300.25	0.47	771.10	2.15	307.81	0.41	767.96	1.92	325.77	0.00
TOU2	940.30	3.40	301.46	0.64	940.47	3.76	321.27	0.66	934.29	3.24	306.77	0.00
TOU3	944.99	3.89	311.27	0.67	943.84	3.25	311.50	0.55	938.66	2.72	325.94	0.00
TOU4	776.73	5.70	268.88	1.04	775.81	4.52	316.39	0.92	768.71	3.63	310.43	0.00
TOU5	800.74	2.93	292.43	0.60	799.79	2.52	312.64	0.48	795.96	2.14	323.48	0.00
TOU6	988.81	4.18	301.59	0.66	988.34	3.74	311.83	0.61	982.30	3.33	324.47	0.00
TOU7	997.05	4.53	284.84	0.79	995.55	4.21	303.13	0.64	989.23	3.51	311.57	0.00
Average	868.11	3.90	294.88	0.73	867.40	3.48	312.03	0.65	**861.84**	**2.96**	318.40	**0.00**

Friedman's test is conducted on TEC_{avg} obtained by TIE, HEA, and ILS on *BCdata*, *BRdata*, and *DPdata*.
The resulting small value of $p = 2.8446e-16 << 0.001$ indicates that the three kinds of optimization frameworks distinguish each other statistically.



(a) TIE on 01a

(b) ILS on 09a

Fig. 6. TIE and ILS with different initial procedures

converges to the same solution quality in the long run. Similar phenomena can be observed for the other instances. This indicates that TIE is not sensitive to the initial starting point. Therefore, we adopt the simple *total_random* initial procedure in this paper.

*2) **Operator TOCX**:* To identify its performance in TIE, we compare the TOCX crossover operator with two other crossover operators of the literature: Precedence preserving order-based crossover (POX) (Park et al., 2021) and path relinking based recombinator (PRX) (Ding et al., 2019). POX is responsible for operation sequencing, along with one-point crossover for machine assignment.

Both POX and PRX are embedded in TIE, and all three versions of TIE are tested on all benchmark instances. Table IX reports detailed results on *BRdata* under TOU5 with statistical analysis. A summary of all results can be found in Table X. In general, TIE with TOCX outperforms PRX and POX in terms of TEC and PRD, and requires slightly more time.

3) ***Neighborhood*** N^t**:** Under the TOU scheme, the new proposed neighborhood N^t plays an important role for TEC minimization. To study the effectiveness of N^t in TIE, we compare TIE (with N^t) and a variant, i.e., TIE without N^t (denoted by TIE$-N^t$), and apply 10 independent runs of each algorithm on all benchmark instances.

Table XI shows detailed results on the *BCdata* while Table XII summarizes the remaining results. We can see that, by integrating the new neighbourhood function N^t, a remarkable reduction on TEC is achieved with negligible additional computational times. Wilcoxon's test also confirms the statistical significance of these results.

4) ***Different evaluation methods*****:** Recall that, among the three evaluation methods, only TEC_exa can generate accurate TEC values for the neighborhood solutions, both TEC_inc and TEC_apx approximate the TEC values since r and q are approximated. For comparison, we apply the evaluation methods on all the instances in *DPdata*, *BCdata*, and *BRdata* with a cutoff time of 10 minutes, and report the computational results in Table XIII. TIE with TEC_apx outperforms the two other evaluation methods, with the smallest values of TEC_{min}, TEC_{avg}, and TEC_{sd}. We conjecture that the approximate evaluation method is computationally efficient while the two other methods suffer from large CPU times. With the same time limit, TIE with TEC_apx can explore a much larger search space. Although the approximate evaluation method may not select the best solution in the neighborhood at each iteration, good solutions are reached in most cases.

In the previous sections, we investigate the performance of the two-individual framework, and examine key components of TIE separately. Test results suggest that:

- A population-based element is necessary to improve the algorithm performance, while focusing on two individuals seem sufficient;
- Sophisticated initial procedures are not necessary for a balanced algorithm, so we can emphasize other components;
- TOCX, which is based on a topological order, can quickly generate offspring solutions with sufficient diversification compared to conventional operators as POX and PRX;
- For solving the additional operation time-tabling subproblem in the FJSP, specific neighbourhood functions such as N^t is beneficial. Considering the comparison with TS(SP), a combination of efficient heuristic and neighbourhood can be desirable;
- Although it does not provide accurate objective values, approximate evaluation methods are useful, especially for this FJSP with large solution spaces.

E. Impact of different TOUs

In this section, we analyze the impact of different TOUs on the performance of TIE which is applied on all the instances with TOU0 to TOU7. For illustration, the boxplot of the resulting TEC in *DPdata*, *BCdata*, and *BRdata* can be found in Fig. 7, where the distributions of TEC with TOU0, TOU1, TOU4, and TOU5 are narrower than with the remaining TOUs, and the values of TEC are smaller as well. It suggests that the average price in TOU contributes to this result. In addition, the range of TEC with TOU0, TOU1, TOU2, and TOU3 are similar to those of TOU4, TOU5, TOU6, and TOU7, respectively. This is probably due to the symmetries of the TOU settings in Table II.

We further explore the impact of different TOU settings by increasing and decreasing the differences of adjacent intervals. In particular, we adopt four additional TOU settings: TOU8 $= \{1, 1.5, 1, 2\}$, TOU9 $= \{1, 3, 1, 5\}$, TOU10 $= \{1, 4, 1, 6\}$, and TOU11 $= \{1, 5, 1, 7\}$, and apply TIE on the 18 instances in *DPdata*. The corresponding results are plotted in Fig. 8.

We can see that the range of TEC increases with the difference of adjacent intervals. However, TIE obtains a relatively smaller makespan with larger differences of adjacent intervals of TOUs (see Fig. 8b). The average required computational time did not change significantly with different TOUs, confirming that TIE is computationally efficient for the considered problem.

From the viewpoint of practitioners, we suggest selecting TOUs with large difference of adjacent intervals to encourage industries to save energy consumption.

IX. Conclusions

In this study, a two-individual-based evolutionary algorithm is proposed to solve the flexible job shop scheduling problem under time-of-use electricity tariffs with the objective of minimizing total energy consumption while satisfying a predefined makespan constraint. To allocate operations to be processed from on-peak periods to off-peak periods, we propose a new neighborhood structure based on varying start times of the operations, along with an approximate neighborhood evaluation method. Based on that, we apply a tabu search procedure to optimize the individuals and a topological order based recombination operator to generate offspring individuals in the evolution process.

Extensive experiments were conducted on well-known benchmark instances, which confirmed the effectiveness of the proposed two-individual structure, and key elements embedded in our algorithm. Based on extensive computational analyzes, we provided some practical suggestions and elements for decision makers to achieve a balance between maximizing productivity and minimizing the electricity cost.

In future work, energy-efficient scheduling in flexible job shop manufacturing systems should be further explored by considering additional practical requirements, such as varying processing speeds and states of machines. In addition to makespan, job completion times, due dates, and other time-based criteria can be formulated as important constraints. Incorporating machine learning approaches into our traditional

TABLE IX
COMPARISON OF DIFFERENT CROSSOVER OPERATORS IN TIE WITH TOU5 ON *BRdata*

Ins.	POX			PRX			TOCX		
	TEC_{min}	TEC_{avg}	TEC_{sd}	TEC_{min}	TEC_{avg}	TEC_{sd}	TEC_{min}	TEC_{avg}	TEC_{sd}
Mk01	31.4	31.5	0.1	31.2	31.32	0.1	30.8	31.04	0.15
Mk02	29	29.12	0.13	29	29.14	0.13	28.8	28.96	0.08
Mk03	157.2	159.86	1.16	157.4	159.21	0.76	157.1	158.93	1.36
Mk04	72.2	72.95	0.38	72.3	72.8	0.22	71.7	72.07	0.24
Mk05	149	149.49	0.31	148.8	149.21	0.25	148.8	149.04	0.23
Mk06	82.3	83.08	0.47	81.6	82.76	0.54	80.9	82.11	0.6
Mk07	156	157.08	0.76	155.1	157.02	0.88	156.1	156.95	0.51
Mk08	459.3	461.93	1.84	458.6	461.84	1.55	452.6	458.03	2.68
Mk09	457.2	460.56	2.13	452.8	458.52	2.77	456.7	460.8	3.08
Mk10	425.1	427.01	1.43	423.8	425.97	1.33	422.8	425.82	1.61
Avg.	201.87	203.26	0.87	201.06	202.78	0.85	**200.63**	**202.38**	1.05

Friedman's test is conducted on TEC_{avg} obtained by TIE, TIE with POX, and TIE with PRX on *BCdata*.
The resulting small value of $p < 0.001$ indicates the three crossover operators distinguish each other statistically.

TABLE X
SUMMARY RESULTS OF DIFFERENT CROSSOVER OPERATORS ON *BCdata*, *BRdata*, AND *DPdata* UNDER TOU0-TOU7

Ins.	POX				PRX				TOCX			
	TEC_{avg}	TEC_{sd}	$time_{avg}$	PRD	TEC_{avg}	TEC_{sd}	$time_{avg}$	PRD	TEC_{avg}	TEC_{sd}	$time_{avg}$	PRD
BCdata	1628.44	6.75	295.58	0.84	1631.15	6.72	292.28	1.00	**1614.96**	**6.00**	301.74	**0.00**
BRdata	209.06	1.43	291.09	0.48	209.83	1.38	293.47	0.85	**208.05**	**1.32**	297.20	**0.00**
DPdata	346.71	**0.75**	335.82	0.01	346.90	0.76	328.35	0.07	**346.66**	**0.75**	342.77	**0.00**

Friedman's test is conducted on TEC_{avg} for all benchmark sets, and shows that the results are statistically significant.

TABLE XI
COMPARISON BETWEEN TIE AND TIE WITHOUT N^t ON *BCdata* UNDER TOU0-TOU3

Ins.	TOU0		TOU1		TOU2		TOU3	
	TIE-N^t TEC_{avg}	TIE TEC_{avg}	TIE-N^t TEC_{avg}	TIE TEC_{avg}	TIE-N^t TEC_{avg}	TIE TEC_{avg}	TIE-N^t TEC_{avg}	TIE TEC_{avg}
mt10c1	932.83	836.70	949.40	890.46	1182.48	1088.79	1193.25	1103.62
mt10cc	921.60	837.51	960.63	893.61	1175.03	1093.17	1160.89	1100.62
mt10x	945.34	853.30	956.97	900.94	1192.08	1107.31	1197.75	1122.36
mt10xx	942.09	851.53	952.98	899.14	1189.99	1104.94	1193.50	1120.46
mt10xxx	943.63	849.77	954.18	898.27	1187.92	1105.66	1192.12	1118.48
mt10xy	925.83	853.52	962.27	906.87	1184.34	1115.86	1178.70	1117.20
mt10xyz	891.47	829.35	958.92	920.78	1156.09	1115.91	1146.76	1092.29
setb4c9	1416.92	1299.80	1463.07	1379.54	1804.89	1694.13	1828.41	1713.57
setb4cc	1408.77	1284.91	1461.42	1374.60	1797.18	1683.69	1812.57	1696.65
setb4x	1413.33	1280.28	1462.80	1368.34	1800.60	1663.71	1822.51	1695.31
setb4xx	1408.43	1279.57	1457.84	1361.32	1791.31	1651.18	1811.97	1684.09
setb4xxx	1406.32	1276.85	1455.18	1361.25	1787.70	1653.35	1812.51	1680.87
setb4xy	1390.83	1268.47	1471.72	1368.64	1791.85	1658.29	1793.27	1679.14
setb4xyz	1368.29	1253.02	1476.87	1362.03	1781.79	1651.88	1776.01	1660.01
seti5c12	2141.26	1943.87	2164.12	2052.95	2685.27	2510.22	2721.74	2507.74
seti5cc	2136.26	1958.95	2189.98	2053.35	2705	2540.71	2709.48	2553.58
seti5x	2119.52	1924.28	2167.04	2059.11	2660.77	2499.02	2681.91	2447.24
seti5xx	2120.41	1923.38	2164.89	2053.22	2658.80	2495.99	2681.92	2445.76
seti5xxx	2123.26	1927.55	2161.61	2052.37	2658.52	2491.38	2680.83	2444.43
seti5xy	2133.12	1964.26	2189.85	2051.82	2700.76	2537.51	2710.47	2556.67
seti5xyz	2107.80	1956.98	2194.01	2055.79	2687.84	2549.94	2693.91	2552.77
Avg.	1485.59	**1354.95**	1532.18	**1441.16**	1884.77	**1762.51**	1895.26	**1766.33**

Wilcoxon signed rank test is conducted on TEC_{avg} for *BCdata*, and shows that the results are statistically significant.

metaheuristics to enhance their performance is also an interesting research direction.

REFERENCES

W.-L. Liu, Y.-J. Gong, W.-N. Chen, Z. Liu, H. Wang, and J. Zhang, "Coordinated charging scheduling of electric vehicles: a mixed-variable differential evolution approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 12, pp. 5094–5109, 2019.

J.-Y. Ding, S. Song, R. Zhang, R. Chiong, and C. Wu, "Parallel machine scheduling under time-of-use electricity prices: New models

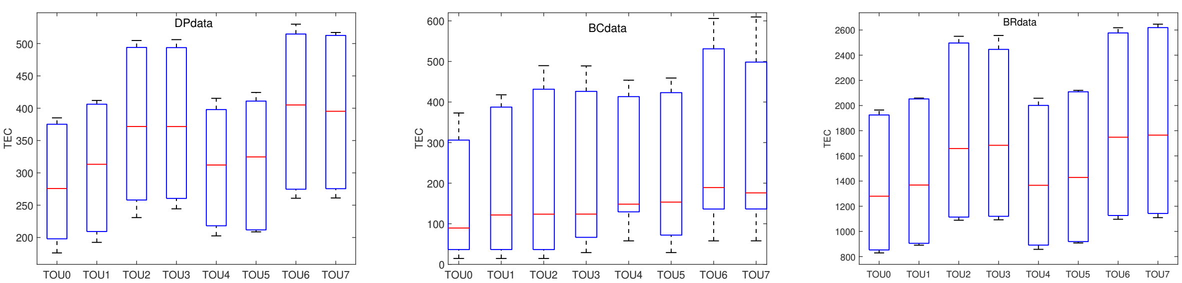
TABLE XII
SUMMARY RESULTS OF TIE $-N^t$ AND TIE ON ALL THE BENCHMARK SETS UNDER TOU0-TOU3

Set	TIE$-N^t$				TIE			
	TEC_{avg}	TEC_{sd}	$time_{avg}$	RPD	TEC_{avg}	TEC_{sd}	$time_{avg}$	RPD
BCdata	1699.45	4.51	304.42	7.48	**1581.24**	**3.67**	301.86	**0.00**
BRdata	187.73	1.29	289.79	6.18	**176.81**	**0.88**	291.41	**0.00**
DPdata	339.09	0.55	357.69	1.08	**335.46**	**0.54**	388.93	**0.00**
Hudata/edata	2045.08	3.55	290.06	3.95	**1967.30**	**3.45**	300.26	**0.00**
Hudata/rdata	2043.24	3.06	295.11	3.01	**1983.45**	**2.60**	298.73	**0.00**
Hudata/sdata	2035.87	4.27	292.37	4.58	**1946.67**	**4.11**	301.18	**0.00**
Hudata/vdata	2043.07	4.14	303.82	3.41	**1975.62**	**3.51**	301.89	**0.00**

Wilcoxon signed rank test is conducted on TEC_{avg} for all benchmark sets, and shows that the results are statistically significant.

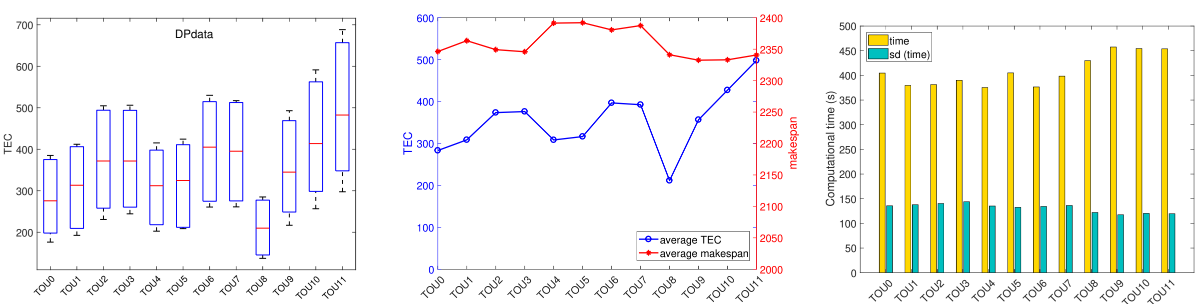
TABLE XIII
SUMMARY RESULTS OF TIE WITH DIFFERENT EVALUATION METHODS ON *DPdata*, *BCdata*, AND *BRdata*

Ins.	TIE with TEC_exa			TIE with TEC_inc			TIE with TEC_apx		
	TEC_{min}	TEC_{avg}	TEC_{sd}	TEC_{min}	TEC_{avg}	TEC_{sd}	TEC_{min}	TEC_{avg}	TEC_{sd}
DPdata	325.15	339.41	7.65	313.29	318.68	2.35	**282.19**	**283.31**	**0.68**
BCdata	174.67	181.50	5.64	153.88	168.51	1.75	**144.03**	**145.60**	**0.85**
BRdata	1487.52	1494.63	10.52	1374.11	1391.29	8.17	**1346.73**	**1354.95**	**4.26**



(a) *DPdata* (b) *BCdata* (c) *BRdata*

Fig. 7. Boxplot of TEC obtained by TIE with different TOU settings



(a) Distribution of TEC (b) Average TEC and makespan (c) Average computational time and its deviation

Fig. 8. Distribution of different features of instances and TIE algorithm

and optimization approaches," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 1138–1154, 2015.

S. Zhou, L. Xing, X. Zheng, N. Du, L. Wang, and Q. Zhang, "A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times," *IEEE transactions on Cybernetics*, vol. 51, no. 3, pp. 1430–1442, 2019.

S. Schulz, U. Buscher, and L. Shen, "Multi-objective hybrid flow shop scheduling with variable discrete production speed levels and time-of-use energy prices," *Zeitschrift für Betriebswirtschaft*, vol. 90, no. 9, pp. 1315–1343, 2020.

L. Shen, S. Dauzère-Pérès, and S. Maecker, "Energy efficient scheduling in flexible job shop manufacturing systems," Working Paper, MSE CMP–SFL 2021/12, Mines Saint-Etienne, Univ Clermont Auvergne, France, 2021.

P. Brucker and R. Schlie, "Job-shop scheduling with multi-purpose machines," *Computing*, vol. 45, no. 4, pp. 369–375, 1990.

C. Özgüven, L. Özbakır, and Y. Yavuz, "Mathematical models for job-shop scheduling problems with routing and process plan flexibility," *Applied Mathematical Modelling*, vol. 34, no. 6, pp. 1539–1548, 2010.

E. G. Birgin, P. Feofiloff, C. G. Fernandes, E. L. De Melo, M. T. Oshiro, and D. P. Ronconi, "A MILP model for an extended version of the flexible job shop problem," *Optimization Letters*, vol. 8, no. 4, pp. 1417–1431, 2014.

R. S. Hansmann, T. Rieger, and U. T. Zimmermann, "Flexible job shop scheduling with blockages," *Mathematical Methods of Operations Research*, vol. 79, no. 2, pp. 135–161, 2014.

M. A. González, C. R. Vela, and R. Varela, "Scatter search with path relinking for the flexible job shop scheduling problem," *European Journal of Operational Research*, vol. 245, no. 1, pp. 35–45, 2015.

S. Kemmoé-Tchomté, D. Lamy, and N. Tchernev, "An effective multi-start multi-level evolutionary local search for the flexible job-shop problem," *Engineering Applications of Artificial Intelligence*, vol. 62, pp. 80–95, 2017.

X. Wu and Y. Sun, "A green scheduling algorithm for flexible job shop with energy-saving measures," *Journal of Cleaner Production*, vol. 172, pp. 3249–3264, 2018.

H. Mokhtari and A. Hasani, "An energy-efficient multi-objective optimization for flexible job-shop scheduling problem," *Computers and Chemical Engineering*, vol. 104, pp. 339–352, 2017.

J.-Q. Li, Y. Du, K.-Z. Gao, P.-Y. Duan, D.-W. Gong, Q.-K. Pan, and P. N. Suganthan, "A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 2153–2170, 2022.

H. Piroozfard, K. Y. Wong, and W. P. Wong, "Minimizing total carbon footprint and total late work criterion in flexible job shop scheduling by using an improved multi-objective genetic algorithm," *Resources, Conservation and Recycling*, vol. 128, pp. 267–283, 2018.

D. Lei, M. Li, and L. Wang, "A two-phase meta-heuristic for multiobjective flexible job shop scheduling problem with total energy consumption threshold," *IEEE Transactions on Cybernetics*, vol. 49, no. 3, pp. 1097–1109, 2019.

H. Zhang, Z. Dai, W. Zhang, S. Zhang, Y. Wang, and R. Liu, "A new energy-aware flexible job shop scheduling method using modified biogeography-based optimization," *Mathematical Problems in Engineering*, vol. 2017, 2017.

E.-D. Jiang and L. Wang, "Multi-objective optimization based on decomposition for flexible job shop scheduling under time-of-use electricity prices," *Knowledge-Based Systems*, vol. 204, p. 106177, 2020.

Y. Du, J.-Q. Li, X.-L. Chen, P.-Y. Duan, and Q.-K. Pan, "Knowledge-based reinforcement learning and estimation of distribution algorithm for flexible job shop scheduling problem," *IEEE Transactions on Emerging Topics in Computational Intelligence*, pp. 1–15, 2022.

L. Chen, N. Megow, R. Rischke, L. Stougie, and J. Verschae, "Optimal algorithms for scheduling under time-of-use tariffs," *Annals of Operations Research*, vol. 304, no. 1, pp. 85–107, 2021.

F. Zhao, X. He, and L. Wang, "A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of no-wait flow-shop problem," *IEEE Transactions on Cybernetics*, vol. 51, no. 11, pp. 5291–5303, 2020.

F. Zhao, L. Zhang, J. Cao, and J. Tang, "A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem," *Computers & Industrial Engineering*, vol. 153, p. 107082, 2021.

F. Zhao, R. Ma, and L. Wang, "A self-learning discrete jaya algorithm for multiobjective energy-efficient distributed no-idle flow-shop scheduling problem in heterogeneous factory system," *IEEE Transactions on Cybernetics*, pp. 1–12, 2021.

C. Y. Zhang, P. G. Li, Z. L. Guan, and Y. Q. Rao, "A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem," *Computers & Operations Research*, vol. 34, no. 11, pp. 3229–3242, 2007.

J. Ding, Z. Lü, C.-M. Li, L. Shen, L. Xu, and F. Glover, "A two-individual based evolutionary algorithm for the flexible job shop scheduling problem," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 2262–2271.

Z. Lü, F. Glover, and J. K. Hao, "A hybrid metaheuristic approach to solving the UBQP problem," *European Journal of Operational Research*, vol. 207, no. 3, pp. 1254–1262, 2010.

A. M. Sutton and F. Neumann, "A parameterized runtime analysis of evolutionary algorithms for the Euclidean traveling salesperson problem," in *AAAI Conference on Artificial Intelligence*, 2012, pp. 595 – 628.

S. Dauzère-Pérès and J. Paulli, "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search," *Annals of Operations Research*, vol. 70, no. 1, pp. 281–306, 1997.

J. W. Barnes and J. B. Chambers, "Flexible job shop scheduling by tabu search." The University of Texas at Austin, Tech. Rep., June 1998.

P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," *Annals of Operations Research*, vol. 41, pp. 57–83, 1993.

J. Hurink, B. Jurisch, and M. Thole, "Tabu search for the job-shop scheduling problem with multi-purpose machines," *Operations-Research-Spektrum*, vol. 15, no. 4, pp. 205–215, 1994.

E. Demirkol, S. Mehta, and R. Uzsoy, "Benchmarks for shop scheduling problems," *European Journal of Operational Research*, vol. 109, no. 1, pp. 137–141, 1998.

L. Shen, S. Dauzère-Pérès, and J. S. Neufeld, "Solving the flexible job shop scheduling problem with sequence-dependent setup times," *European Journal of Operational Research*, vol. 265, pp. 503–516, 2018.

J. Ding, Z. Lü, T. Cheng, and L. Xu, "A hybrid evolutionary approach for the single-machine total weighted tardiness problem," *Computers & Industrial Engineering*, vol. 108, pp. 70–80, 2017.

F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in Statistics*. Springer, 1992, pp. 196–202.

J.-S. Park, H.-Y. Ng, T.-J. Chua, Y.-T. Ng, and J.-W. Kim, "Unified genetic algorithm approach for solving flexible job-shop scheduling problem," *Applied Sciences*, vol. 11, no. 14, p. 6454, 2021.