2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Local Search Enhanced Multi-objective Evolutionary Algorithm for Fuzzy Flexible Job Shop Scheduling

Xuwei Zhang, Ziyan Zhao, Shixin Liu
The State Key Laboratory of Synthetical Automation for Process Industries
College of Information Science and Engineering
Northeastern University
Shenyang, 110819, China
xuwei_zhang163@163.com, zhaoziyan@mail.neu.edu.cn, sxliu@mail.neu.edu.cn

***Abstract*—The uncertainty in actual manufacturing systems often manifests as uncertain processing times, especially in flexible manufacturing systems. This work proposes a Decomposition-based Evolutionary Algorithm with Local Search (DLSEA) to solve flexible scheduling with fuzzy processing times by minimizing makespan and total machine workload. Considering the different scales of objectives, two normalization methods are employed on subpopulations, respectively, aiming to mitigate the potential detrimental effects of a single normalization method. This work also introduces a local search method to enhance the performance of DLSEA. The proposed DLSEA is compared with four state-of-the-art algorithms on two series of cases. The experimental results show that DLSEA exhibits superior search capabilities.**

***Index Terms*—Fuzzy flexible job shop scheduling, Normalization, Local search, Multi-objective evolutionary algorithm**

I. Introduction

In practical manufacturing systems, some inevitable situations cause fluctuations in job processing time within a certain range [1], presenting uncertainty [2]. Triangular fuzzy numbers [3]–[5] have attracted widespread attention due to their ability to simulate uncertain processing time in flexible job-shop scheduling problems (FJSPs) [6]–[8]. It predicts the most likely processing time through fuzzy membership function.

FJSPs need to specify a suitable processing machine for each task to minimize or maximize a certain objective. It is considered as an NP-hard problem, which makes it challenging to solve in polynomial time [9]. Consequently, a large number of approximation algorithms have been proposed to tackle the FJSPs, such as genetic algorithm [10], ant colony optimization [11], artificial bee colony algorithm [12], grey wolf optimizition [13], differential evolution [14], brain storm optimization algorithm [15] and teaching-learning based optimization [16].

Most of the aforementioned work only considers a single objective, such as makespan, and energy consumption. However, in actual manufacturing systems, it is often necessary to simultaneously consider multiple conflicting objectives [17]. Consequently, the research focus shifts to the multi-objective flexible job shop scheduling problems with fuzzy processing time (MFFJSPs) [18]–[20].

Zhong et al. [8] propose a modified artificial bee colony algorithm (MABC) to solve MFFJSP that minimizes makespan, weighted agreement index and maximum machine workload. In MABC, three-point satisfaction-degree model and local search based on variable neighborhood search are employed to enhance the performance of MABC. Wang et al. [19] develop a fast non-dominated sorting genetic algorithm II (NSGA-II [21]) embedded with local simulated-annealing operators to solve MFFJSP. A bi-population evolutionary algorithm that adjusts the size of subpopulations through a feedback mechanism (FBEA) is proposed in [22]. Li et al. [23] propose an improved artificial immune system algorithm to handle MFFJSP considering makespan and energy consumption. In it, six local search approaches and a simulated annealing method are embedded to enhance its exploration abilities. In addition, a novel population diversity heuristic is presented to eliminate antibodies with high crowding values. Piroozfard et al. [24] develop an improved multi-objective genetic algorithm to minimize total carbon footprint and total late work criterion in MFFJSP. From an evolutionary mechanism perspective, the above algorithms can be divided into the same kind of algorithms, that is, dominance-based multi-objective evolutionary algorithm. However, these algorithms often require the calculation of diversity evaluation indicators to maintain population diversity. This may not apply to MFFJSPs. Decomposition-based multi-objective algorithm (MOEA/D) [25] maintains population diversity through preset reference vectors in the objective space without using diversity indicators. Li et al. [26] propose a hybrid adaptive parameter evolutionary algorithm based on decomposition (HPEA) to solve MFFJSP, considering makespan and total workload. A reinforcement learning based MOEA/D (RMOEA/D) is developed in literature [27]. In it, a parameter adaptive strategy based on Q-learning is employed to assist the algorithm in choosing the best parameters. Liu et al. [28] propose an improved MOEA/D embedded with neighborhood search to minimize makespan, total machine load, and maximum machine load in MFFJSP.

Although the aforementioned work demonstrates promising performance in addressing MFFJSP, the scales of different

978-0-7381-4408-5/23/$31.00 ©2023 IEEE

objectives often varies. It may cause the algorithm to have different levels of attention to different objectives. This work proposes a decomposition-based evolutionary algorithm with local search (DLSEA) to solve MFFJSP. The main contributions are as follows:

1) Two normalization methods are separately used for two co-evolutionary subpopulations to balance the impact of different objective scales on the algorithm.
2) An adaptive local search method is designed to enhance the convergence ability of the algorithm.

Fuzzy operators and problem statement are given in Section II. The proposed DLSEA is developed in Section III. Section IV shows the experimental results. Conclusions and future work are given in Section V.

II. Fuzzy operators and the MFFJSP

A. Fuzzy operators

Triangular fuzzy number (TFN) [29] is used to simulate uncertain processing times in this work. For two TFNs $\tilde{z} = (z_1, z_2, z_3)$ and $\tilde{t} = (t_1, t_2, t_3)$, addition and rank operator are defined as follows.

1) Addition operator:

$$\tilde{z} + \tilde{t} = (z_1 + t_1, z_2 + t_2, z_3 + t_3) \tag{1}$$

2) Rank operator:

$$I_1(\tilde{t}) = \frac{t_1 + 2t_2 + t_3}{4} \tag{2}$$

Condition 1: if $I_1(\tilde{z}) > I(\tilde{t})$, then $\tilde{z} > \tilde{t}$. Otherwise $\tilde{z} < \tilde{t}$.

Condition 2: when $I_1(\tilde{z}) = I_1(\tilde{t})$, $I_2(\tilde{t}) = t_2$. If $I_2(\tilde{z}) > I_2(\tilde{t})$, then $\tilde{z} > \tilde{t}$. Otherwise $\tilde{z} < \tilde{t}$.

Condition 3: when $I_2(\tilde{z}) = I_2(\tilde{t})$, $I_3(\tilde{t}) = t_3 - t_1$. If $I_3(\tilde{z}) > I_3(\tilde{t})$, then $\tilde{z} > \tilde{t}$. Otherwise $\tilde{z} < \tilde{t}$.

B. Problem Description

MFFJSP requires allocating appropriate processing machines for tasks to minimize or maximize a certain objective. Assume a set of n jobs, $J = \{J_1, J_2, \ldots, J_n\}$, and a set of m machines $M = \{M_1, M_2, \ldots, M_m\}$. And $J_i = \{O_{i,1}, O_{i,2}, \ldots, O_{i,h_i}\}$, where h_i represents the operations number of J_i. The processing time of operation $O_{i,j}$ on machine M_k is a TFN $\tilde{p} = (p_1, p_2, p_3)$. In this work, makespan and total workload are considered. Makespan refers to the maximum completion time of a job and measures production efficiency to a certain extent. Total workload reflects the degree of machine wear. Makespan prioritizes allocating operations to machines with fast processing times, which may lead to an increase in the total workload. Consequently, there is a conflict between the makespan and total workload objectives.

$$Makespan = max\{\tilde{c}_k\} \tag{3}$$

where $\tilde{c}_k$ is the completion time of the last operation in scheduling sequence.

$$Total\ workload = \sum_k \sum_i \sum_j \tilde{p}_{i,j,k} \cdot d_{i,j,k} \tag{4}$$

where $\tilde{p}_{i,j,k}$ is the fuzzy processing time of operation $O_{i,j}$. $d_{i,j,k}$ is a 0-1 notation that represents whether the operation $O_{i,j}$ is processed on machine M_k.

The details of a mixed integer linear programming model of MFFJSP can be seen in [26].

III. Proposed algorithm

In order to solve MFFJSP, a decomposition-based evolutionary algorithm with local search is developed in this section.

The pseudocode of DLSEA is presented in Algorithm 1. In the initialization stage, the reference vectors are preset in the objective space, and then population P_1 and P_2 are initialized. A two-layer coding scheme [30] is used to represent each solution. Two normalization methods are utilized in P_1 and P_2, respectively. After the offspring are generated, the elite solutions are retained. Finally, local search is performed on P_1 and P_2.

A. Initialization

In this work, the solution of two-layer encoding is randomly initialized in P_1, and P_2 is cloned from P_1.

Initialization steps: Randomly rearrange all operations and combine them into a scheduling sequence vector. Then, randomly assign a candidate machine for each operation and combine them into a machine sequence vector.

B. Evolutionary mechanism

In DLSEA, for a parent solution x, another parent solution is selected based on the tournament. Afterwards, precedence operation crossover (POX) [31] and universal crossover (UX) [31] are used to generate offspring. POX and UX act on the operation code and machine code, respectively. There are three types of mutation operators: swap, insert, and inverse. During algorithm execution, a mutation operator is randomly selected and executed.

To ensure equal emphasis on each objective, P_1 and P_2 respectively perform different normalization methods. Among them, a non-standard normalization method [32] is implemented in P_1. Its definition formula is as follows:

$$f_i^{'}(x) = f_i(x) - z^* \tag{5}$$

where $z^* = min(f_i^1(x), f_i^2(x), \ldots, f_i^m(x))$, m is the number of objectives. z^* is the ideal point. f_i is the objective vector, and $i = 1, 2, \ldots, N$.

Another widely adopted normalization method [33] is implemented in P_2. This approach is grounded upon the notion of ideal and nadir points, and its definition formula is as follows:

$$f_i^{'}(x) = \frac{f_i(x) - z^*}{z^n - z^* + \varepsilon} \tag{6}$$

where $z^n = max(f_i^1(x), f_i^2(x), \ldots, f_i^m(x))$ is the nadir point. ε is a sufficiently number, it is set to $1e-6$ in this work.

A tchebycheff decomposition method [25] is employed to transform MFFJSP into multiple single objective problems. It maintains population diversity through preset reference vectors and is defined as follows:

$$g(x \mid \lambda) = \max_{1 \le i \le m} \{\lambda_i \mid f_i(x) - z^* \mid\} \tag{7}$$

where λ_i is a reference vector. And the size of reference vector is N.

Algorithm 1 The pseudocode of proposed DLSEA.

Input: Population size: N; Mutation rate ρ; Maximum number of function evaluations (Fes_max);
Output: Non-dominant solutions;
Initialize: Reference Vector (λ);
$P_1 \leftarrow$ Random initialization;
$P_2 \leftarrow P_1$;
while $Fes < Fes\_max$ **do**
  % Generate offspring;
  for $j = 1 : N/2$ **do**
    x_j^1 and x_j^2 from P_1 and P_2, respectively;
    x_k and $x_l \leftarrow$ by binary tournament from P_1 and P_2 respectively;
    $y_1 \leftarrow$ by cross and mutation operators (x_j^1,x_k);
    $y_2 \leftarrow$ by cross and mutation operators (x_j^2,x_l);
    Update ideal point and nadir point;
    % Population P_1 evolution;
    $g(x_j^1) \leftarrow max\{\lambda_j \mid f(y_1) - z^*\}$;
    Similarly, $g(y_1)$ and $g(y_2)$ are obtained by Eq.(7);
    if $min[g(x_j^1), g(y_1), g(y_2)] = g(y_1)$ **then**
      $x_j^1 = y_1$
    else if $min[g(x_j^1), g(y_1), g(y_2)] = g(y_2)$ **then**
      $x_j^2 = y_2$
    end if
    % Population P_2 evolution;
    $g(x_j^2) \leftarrow max\{\lambda_j \mid \frac{f(x_j^2) - z^*}{z^n - z^*}\}$;
    Similarly, $g(y_1)$ and $g(y_2)$ are obtained by Eq.(7);
    if $min[g(x_j^2), g(y_1), g(y_2)] = g(y_1)$ **then**
      $x_j^2 = y_1$
    else if $min[g(x_j^1), g(y_1), g(y_2)] = g(y_2)$ **then**
      $x_j^2 = y_2$
    end if
    $Fes = Fes + 2$
  end for
  Local search for P_1 and P_2 by Algorithm 2
end while

C. Local search

In this section, an adaptive local search method is proposed. This method is embedded with three local search operators, which are applied to the current solution to enhance the convergence ability of the algorithm. The local search operators are adaptively selected based on success and failure rates. The details of the three local search operators are given as follows:

operator 1: Determine the last completed operation and assign it to the fastest processing machine.

operator 2: Randomly select an operation and move it to the fastest processing machine.

operator 3: Determine the machine M with the highest load and randomly move an operation processed by M to other machine.

The execution details of local search are shown in Algorithm 2. The number of successes and failures of each operator in each iteration are recorded in S and F, respectively. S and F only record data from the last L twenty iterations, where $L = 10$. Then the probabilities p_i are calculated. Finally, an operator is adaptively selected and executed according to roulette wheel selection method. Note that local search is executed in the later stage of DLSEA, and a triggering condition is defined as: $Fes \ge 0.8 \cdot Fes\_Max$ [34].

Algorithm 2 The pseudocode of local search.

Input: Population: P_1 and P_2;
Output: Population: P_1 and P_2;
% Calculate the probability of each operator
if size(S,2)$\ge L$ **then**
  $p(i) = \frac{sum(S(i,:))}{sum(S(i,:)) + sum(F(i,:))}$
  S and F records the most recent L times of data;
else
  $p(i) = 1/3,\ i = 1, 2, 3$;
end if
$p(i) = \frac{p(i)}{\sum p(i)},\ i = 1, 2, 3$;
for $j = 1 : N/2$ **do**
  $\kappa \leftarrow$ Roulette $(p(1), p(2), p(3))$;
  u_j^1 and u_j^2 from P_1 and P_2, respectively;
  $v_1 \leftarrow$ operator κ (u_j^1)
  $v_2 \leftarrow$ operator κ (u_j^2)
  $g(v_1)$ and $g(u_j^1)$ are obtained by Eq.(7) and (5);
  if $g(v_1) < g(u_j^1)$ **then**
    $u_j^1 = v_1$
    $S(\kappa, t) = S(\kappa, t) + 1$; %Record Success Times
  else
    $F(\kappa, t) = F(\kappa, t) + 1$; % Record failures Times
  end if
  $g(v_2)$ and $g(u_j^2)$ are obtained by Eq.(7) and (5);
  if $g(v_2) < g(u_j^2)$ **then**
    $u_j^2 = v_2$
    $S(\kappa, t) = S(\kappa, t) + 1$; %Record Success Times
  else
    $F(\kappa, t) = F(\kappa, t) + 1$; % Record failures Times
  end if
end for

IV. Experimental studies

In this section, the performance of DLSEA is empirically examined on two widely used benchmark case FMk [26] and remanu [12]. DLSEA is compared with two widely-accepted

TABLE I: HV values obtained by DLSEA, HPEA, MOEAD_M2M, NSGA-II and MOEA/D.

	DLSEA	HPEA	MOEAD_M2M	NSGA-II	MOEA/D
remanu01	3.1047E-02(2.87E-03)	**3.2657E-02(2.13E-03)=**	2.4414E-02(6.82E-03)+	2.9646E-02(2.55E-03)+	3.2280E-02(2.49E-03)=
remanu02	1.6377E-01(2.93E-03)	**1.6471E-01(3.79E-03)=**	4.1653E-02(2.10E-02)+	1.4510E-01(6.12E-03)+	1.6126E-01(4.65E-03)+
remanu03	**1.5002E-01(2.32E-03)**	1.4757E-01(2.33E-03)+	4.8529E-02(1.20E-02)+	1.4218E-01(4.94E-02)+	1.4320E-01(1.95E-03)+
remanu04	**2.0753E-01(4.15E-03)**	1.9730E-01(7.81E-03)+	6.2824E-02(7.01E-03)+	1.6825E-01(6.17E-03)+	1.9542E-01(3.80E-03)+
remanu05	1.8433E-01(5.26E-03)	**2.0262E-01(6.75E-02)-**	2.6496E-02(9.36E-03)+	1.6336E-01(5.70E-02)+	1.9850E-01(7.49E-02)=
remanu06	**1.9860E-01(4.19E-03)**	1.8894E-01(6.14E-03)+	1.7290E-02(1.04E-02)+	1.5411E-01(1.16E-02)+	1.8753E-01(3.38E-03)+
remanu07	**2.4793E-01(6.80E-02)**	2.2776E-01(4.96E-03)+	2.1772E-02(9.39E-03)+	2.0704E-01(6.49E-02)+	2.3783E-01(7.04E-02)+
remanu08	**3.8976E-01(4.09E-03)**	3.5793E-01(1.39E-02)+	5.1467E-02(8.97E-03)+	3.2661E-01(1.21E-02)+	3.6868E-01(1.34E-02)+
FMk01	6.7863E-02(2.09E-03)	7.2144E-02(1.90E-03)-	5.5340E-02(8.55E-03)=	6.5029E-02(3.63E-03)+	**7.3928E-02(2.58E-03)-**
FMk02	1.3058E-01(4.82E-03)	**1.3746E-01(2.93E-03)=**	5.7249E-02(8.34E-03)+	1.2440E-01(2.75E-03)+	1.3124E-01(2.00E-03)=
FMk03	1.4850E-01(8.18E-03)	1.5364E-01(1.18E-02)=	3.2915E-02(1.31E-02)+	**1.8599E-01(2.02E-01)-**	1.5862E-01(8.87E-03)-
FMk04	1.4213E-01(4.61E-03)	1.5155E-01(3.76E-03)=	1.2417E-01(6.30E-03)+	1.0655E-01(1.26E-02)+	**1.5733E-01(3.61E-03)=**
FMk05	**6.2399E-02(7.30E-02)**	4.3837E-02(1.74E-03)+	3.2524E-02(2.21E-03)+	3.8230E-02(2.52E-03)+	4.3892E-02(1.40E-03)+
FMk06	2.5036E-01(4.00E-03)	**2.7169E-01(6.09E-03)=**	6.9637E-02(2.33E-02)+	2.4694E-01(8.51E-03)=	2.6628E-01(6.36E-03)=
FMk07	**1.3597E-01(2.94E-03)**	1.1288E-01(8.68E-03)+	3.6276E-02(1.19E-02)+	9.7071E-02(4.45E-03)+	1.1253E-01(5.19E-03)+
FMk08	**1.2580E-01(2.04E-01)**	3.0768E-02(1.81E-03)+	1.7824E-02(2.34E-03)+	7.4384E-02(1.53E-01)+	3.2624E-02(1.94E-03)+
FMk09	7.3529E-02(5.15E-03)	**8.0654E-02(5.82E-03)=**	6.5149E-02(5.15E-03)=	6.6212E-02(3.84E-03)+	7.9750E-02(2.27E-03)-
FMk010	8.1381E-02(3.40E-03)	**8.7675E-02(4.57E-03)-**	6.0396E-02(6.34E-03)+	7.0363E-02(5.59E-03)+	8.5126E-02(2.39E-03)=
	+/=/-	10/5/3	16/2/0	16/1/1	9/6/3

TABLE II: GD values obtained by DLSEA, HPEA, MOEAD_M2M, NSGA-II and MOEA/D.

	DLSEA	HPEA	MOEAD_M2M	NSGA-II	MOEA/D
remanu01	9.0273E-03(1.32E-02)	**5.1969E-03(5.27E-03)-**	6.0197E-02(6.60E-02)+	3.1459E-02(3.61E-02)+	7.7946E-03(9.63E-03)-
remanu02	**1.8325E-02(1.07E-02)**	2.2725E-02(1.52E-02)+	1.8170E-01(4.50E-02)+	6.2016E-02(1.35E-02)+	2.5299E-02(9.21E-03)+
remanu03	1.9004E-01(4.20E-02)	1.4257E-01(2.61E-02)-	2.3206E-01(4.27E-02)+	**1.2642E-01(4.89E-02)-**	1.4735E-01(3.77E-02)=
remanu04	**5.9238E-03(2.52E-03)**	2.1511E-02(1.89E-02)+	1.7936E-01(3.46E-02)+	4.7464E-02(9.43E-03)+	2.0481E-02(5.12E-03)+
remanu05	1.7150E-01(3.05E-02)	**1.5034E-01(6.02E-02)=**	3.6993E-01(6.93E-02)+	1.8210E-01(4.62E-02)+	1.6151E-01(8.27E-02)=
remanu06	**1.1256E-02(7.32E-03)**	2.3461E-02(7.18E-03)+	2.1566E-01(6.97E-02)+	5.8138E-02(7.04E-03)+	2.2186E-02(7.54E-03)+
remanu07	1.5357E-01(6.63E-02)	**1.2706E-01(2.08E-02)-**	3.1046E-01(6.26E-02)+	1.3472E-01(3.69E-02)=	1.3109E-01(6.16E-02)=
remanu08	**5.3272E-03(4.34E-03)**	1.6622E-02(4.84E-03)+	2.2492E-01(2.27E-02)+	3.1691E-02(4.02E-03)+	1.8987E-02(4.07E-03)+
FMk01	2.2214E-02(2.36E-02)	2.7186E-02(1.02E-02)+	6.3507E-02(1.16E-02)=	3.3933E-02(1.89E-02)=	**1.2793E-02(7.34E-03)-**
FMk02	**1.0339E-02(5.07E-03)**	1.0639E-02(5.22E-03)+	1.1620E-01(1.60E-02)+	3.3933E-02(1.04E-02)+	1.1160E-02(3.35E-03)+
FMk03	2.4849E-01(2.48E-02)	2.3062E-01(3.62E-02)=	5.5056E-01(7.78E-02)+	2.5865E-01(9.31E-02)=	**1.9788E-01(1.36E-02)-**
FMk04	6.0294E-03(1.61E-03)	**2.3968E-03(6.81E-04)-**	2.1918E-02(3.02E-03)+	6.7856E-03(7.42E-03)+	2.4726E-03(4.28E-04)+
FMk05	**1.5632E-01(5.63E-02)**	1.5919E-01(1.92E-02)+	2.2518E-01(1.45E-02)+	1.7283E-01(1.91E-02)+	1.6668E-01(1.98E-02)+
FMk06	2.5923E-02(5.91E-03)	**9.8300E-03(6.17E-03)-**	1.8133E-01(4.71E-02)+	4.0396E-02(6.87E-03)+	1.5400E-02(5.61E-03)=
FMk07	1.7201E-01(1.40E-02)	1.3678E-01(1.71E-02)=	2.4568E-01(3.98E-02)+	1.4725E-01(9.59E-03)-	**1.2966E-01(1.57E-02)-**
FMk08	**2.7365E-01(1.46E-01)**	3.9236E-01(5.78E-02)+	5.2340E-01(1.03E-01)+	2.8018E-01(1.01E-01)=	2.9143E-01(4.55E-02)+
FMk09	1.7250E-02(8.09E-03)	**5.1320E-03(4.06E-03)-**	6.9721E-02(1.11E-02)+	3.0936E-02(9.99E-03)+	1.1196E-02(3.63E-03)=
FMk010	**8.2372E-03(2.29E-03)**	1.0396E-02(7.43E-03)+	1.0307E-01(1.38E-02)+	2.9284E-02(3.75E-03)+	1.1422E-02(6.07E-03)+
	+/=/-	9/3/6	17/1/0	12/2/4	9/5/4

evolutionary algorithm, namely, NSGA-II [21] and MOEA/D [25], and two state-of-the-art evolutionary algorithm, i.e., HPEA [26] and MOEAD_M2M [20]. The parameters for all compared algorithms are consistent with the original literature and shown in Table III.

TABLE III: The parameter settings of all algorithms.

Algorithm	Parameter information
MOEA/D [25]	mutation rate=0.8, T=10 .
NSGA-II [21]	mutation rate=0.8.
MOEAD_M2M [20]	mutation rate=0.8, K=10.
HPEA [26]	Lp=45, T=$\{3, 5, 7, 8, 10, 12\}$, mutation rate=0.8.
DLSEA	mutation rate=0.8, $L = 10$

For fairness, all algorithms run 20 independent times on two widely used benchmark cases, and the population size is set to 100. The maximum calculation number of fitness values is the termination condition of all algorithms, and the maximum calculation number of fitness values is set to 10000. The code platform and execution environment of the algorithms are MATLAB 2020b and Intel(R) Core(TM) i5-1135G7 @2.40GHz with 16G RAM. Hypervolume (HV) [35] and Generation Distance (GD) [36] are employed to reflect the performance of the algorithm.

The HV and GD values obtained by DLSEA, HPEA, MOEAD_M2M, NSGA-II, and MOEA/D are shown in Tables I and II, respectively. The non-dominated solutions obtained by all algorithms on FMk04, remanu04, and remanu06 are presented in Figs 1, 2, and 3, respectively. From the above figure, it can be seen that non-dominated solutions obtained by DLSEA have good convergence, but their number is relatively small. This reflects the shortcomings of DLSEA in maintaining population diversity.

For HV, it can be seen that the HV values obtained by DLSEA are superior to HPEA, MOEAD_M2M, NSGA-II, and

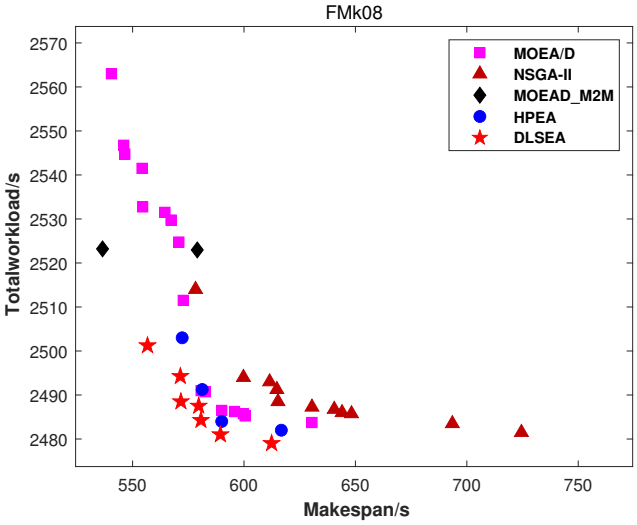

Fig. 1: Non-dominated solutions obtained by all algorithms on FMk08.

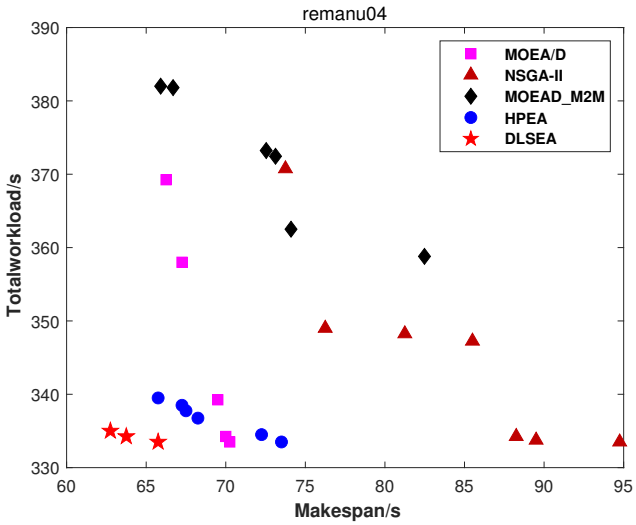

Fig. 2: Non-dominated solutions obtained by all algorithms on remanu04.

MOEA/D on most cases. Specifically, DLSEA outperforms other algorithms on 8 out of 18 cases, HPEA outperforms other algorithms on 6 out of 18 cases, and MOEA/D outperforms other algorithms on 2 out of 18 cases. Although the performance advantage of DLAEA is not significant compared to HPEA, it is still superior to HPEA. In addition, $+$, $=$, and $-$ represents different results of the wilcoxon signed rank test ($\alpha = 0.05$). The statistical analysis results also indicate that the performance of DLSEA is superior to other comparative algorithms.

For GD, it can be seen that the GD values obtained by DLSEA are superior to HPEA, MOEAD_M2M, NSGA-II, and MOEA/D on most cases. Specifically, DLSEA outperforms other algorithms on 9 out of 18 cases, HPEA outperforms other algorithms on 6 out of 18 cases, and MOEA/D outperforms other algorithms on 3 out of 18 cases. Obviously, the performance of DLSEA is not significant compared to HPEA, but it is still superior to HPEA. In addition, from the results of statistical analysis, DLAEA outperformed HPEA on 9 out of 18 cases, and there is no significant difference on 3 cases compared to HPEA, only weaker than HPEA on 6 out of 18 cases.

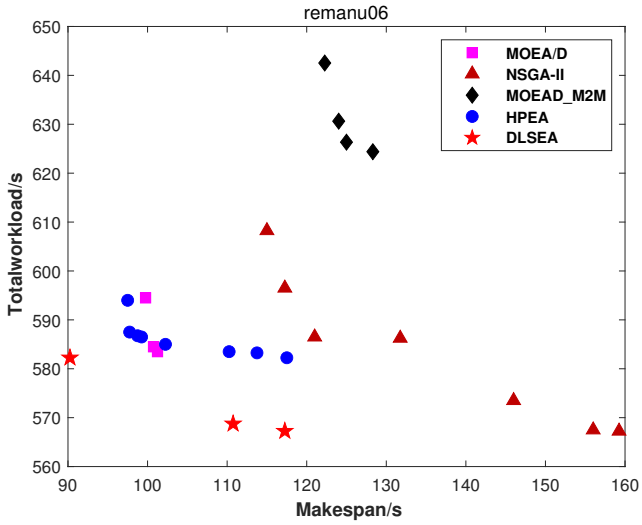

Fig. 3: Non-dominated solutions obtained by all algorithms on remanu06.

V. Conclusion

In this work, a decomposition-based evolutionary algorithm with local search (DLSEA) is developed to solve MFFJSP, which aims to minimize makespan and total machine workload. To account for the different scales of these objectives, two normalization methods are applied to each subpopulation, ensuring equal emphasis on each objective. This work also designs an adaptive local search method that selects a suitable local search operator based on the evolutionary state, thereby enhancing the convergence of the DLSEA. To evaluate the performance of DLSEA, empirical comparisons are conducted by comparing DLSEA with four state-of-the-art algorithms. The experimental results demonstrate that the performance of DLSEA is competitive. In the future, we will expand DLSEA to address MFFJSP with machine failure.

Acknowledgment

The authors would like to thank the National Key R&D Program of China under Grant No. 2021YFB3301200, and National Natural Science Foundation of China under Grant No. 62073069 and 62203093, and LiaoNing Revitalization Talents Program under Grant No. XLYC2002041, and Fundamental Research Funds for the Central Universities under Grant No. N2204016.

REFERENCES

[1] J. Lin, "A hybrid biogeography-based optimization for the fuzzy flexible job-shop scheduling problem," *Knowledge-Based Systems*, vol. 78, pp. 59–74, 2015.

[2] R. Li, W. Gong, L. Wang, C. Lu, and S. Jiang, "Two-stage knowledge-driven evolutionary algorithm for distributed green flexible job shop scheduling with type-2 fuzzy processing time," *Swarm and Evolutionary Computation*, vol. 74, p. 101139, 2022.

[3] D. Lei, "A genetic algorithm for flexible job shop scheduling with fuzzy processing time," *International Journal of Production Research*, vol. 48, no. 10, pp. 2995–3013, 2010.

[4] D. Lei, "Co-evolutionary genetic algorithm for fuzzy flexible job shop scheduling," *Applied Soft Computing*, vol. 12, no. 8, pp. 2237–2245, 2012.

[5] J. Lin, "Backtracking search based hyper-heuristic for the flexible job-shop scheduling problem with fuzzy processing time," *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 186–196, 2019.

[6] L. Sun, L. Lin, M. Gen, and H. Li, "A hybrid cooperative coevolution algorithm for fuzzy flexible job shop scheduling," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 5, pp. 1008–1022, 2019.

[7] Y. Dorfeshan, R. Tavakkoli-Moghaddam, S. M. Mousavi, and B. Vahedi-Nouri, "A new weighted distance-based approximation methodology for flow shop scheduling group decisions under the interval-valued fuzzy processing time," *Applied Soft Computing*, vol. 91, p. 106248, 2020.

[8] Z. Yuguang, Y. Fan, and L. Feng, "Solving multi-objective fuzzy flexible job shop scheduling problem using MABC algorithm," *Journal of Intelligent & Fuzzy Systems*, vol. 36, no. 2, pp. 1455–1473, 2019.

[9] Z. Zhao, M. Zhou, and S. Liu, "Iterated greedy algorithms for flow-shop scheduling problems: A tutorial," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1941–1959, 2021.

[10] M. Sakawa and T. Mori, "An efficient genetic algorithm for job-shop scheduling problems with fuzzy processing time and fuzzy duedate," *Computers & Industrial Engineering*, vol. 36, no. 2, pp. 325–341, 1999.

[11] Z. Jia, J. Yan, J. Y. Leung, K. Li, and H. Chen, "Ant colony optimization algorithm for scheduling jobs with fuzzy processing time on parallel batch machines with different capacities," *Applied Soft Computing*, vol. 75, pp. 548–561, 2019.

[12] K. Z. Gao, P. N. Suganthan, T. J. Chua, C. S. Chong, T. X. Cai, and Q. K. Pan, "A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7652–7663, 2015.

[13] X. Guo, Z. Zhang, L. Qi, S. Liu, Y. Tang, and Z. Zhao, "Stochastic hybrid discrete grey wolf optimizer for multi-objective disassembly sequencing and line balancing planning in disassembling multiple products," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1744–1756, 2021.

[14] D. Gao, G. Wang, and W. Pedrycz, "Solving fuzzy job-shop scheduling problem using DE algorithm improved by a selection mechanism," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 12, pp. 3265–3275, 2020.

[15] J. Li, J. Li, L. Zhang, H. Sang, Y. Han, and Q. Chen, "Solving type-2 fuzzy distributed hybrid flowshop scheduling using an improved brain storm optimization algorithm," *International Journal of Fuzzy Systems*, vol. 23, pp. 1194–1212, 2021.

[16] Y. Xu, L. Wang, S. Wang, and M. Liu, "An effective teaching–learning-based optimization algorithm for the flexible job-shop scheduling problem with fuzzy processing time," *Neurocomputing*, vol. 148, pp. 260–268, 2015.

[17] Z. Zhao, S. Liu, M. Zhou, and A. Abusorrah, "Dual-objective mixed integer linear program and memetic algorithm for an industrial group scheduling problem," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 6, pp. 1199–1209, 2020.

[18] J. J. Palacios, I. Gonzalez-Rodriguez, C. R. Vela, and J. Puente, "Robust multiobjective optimisation for fuzzy job shop problems," *Applied Soft Computing*, vol. 56, pp. 604–616, 2017.

[19] B. Wang, H. Xie, X. Xia, and X. Zhang, "A NSGA-II algorithm hybridizing local simulated-annealing operators for a bi-criteria robust job-shop scheduling problem under scenarios," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 5, pp. 1075–1084, 2018.

[21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[22] Z. Pan, D. Lei, and L. Wang, "A bi-population evolutionary algorithm with feedback for energy-efficient fuzzy flexible job shop scheduling," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 8, pp. 5295–5307, 2022.

[23] J. Li, Z. Liu, C. Li, and Z. Zheng, "Improved artificial immune system algorithm for type-2 fuzzy flexible job shop scheduling problem," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 11, pp. 3234–3248, 2020.

[24] 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.

[25] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[26] R. Li, W. Gong, and C. Lu, "Self-adaptive multi-objective evolutionary algorithm for flexible job shop scheduling with fuzzy processing time," *Computers & Industrial Engineering*, vol. 168, p. 108099, 2022.

[27] R. Li, W. Gong, and C. Lu, "A reinforcement learning based RMOEA/D for bi-objective fuzzy flexible job shop scheduling," *Expert Systems with Applications*, vol. 203, p. 117380, 2022.

[28] Z. Liu, X. Liang, L. Hou, D. Yang, and Q. Tong, "Multi-strategy dynamic evolution-based improved MOEA/D algorithm for solving multi-objective fuzzy flexible job shop scheduling problem," *IEEE Access*, vol. 11, pp. 54 596–54 606, 2023.

[29] M. Sun, Z. Cai, and H. Zhang, "A teaching-learning-based optimization with feedback for LR fuzzy flexible assembly job shop scheduling problem with batch splitting," *Expert Systems with Applications*, vol. 224, p. 120043, 2023.

[30] B. Wei, J. Qi, and J. Wen, "Research on job-shop scheduling problem based-on improved genetic algorithm," in *Journal of Physics: Conference Series*, vol. 2216, no. 1. IOP Publishing, 2022, p. 012083.

[31] R. Li, W. Gong, C. Lu, and L. Wang, "A learning-based memetic algorithm for energy-efficient flexible job-shop scheduling with type-2 fuzzy processing time," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 3, pp. 610–620, 2023.

[32] C. Liu, Q. Zhao, B. Yan, S. Elsayed, T. Ray, and R. Sarker, "Adaptive sorting-based evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 247–257, 2018.

[33] H. Ishibuchi, K. Doi, and Y. Nojima, "On the effect of normalization in MOEA/D for multi-objective and many-objective optimization," *Complex & Intelligent Systems*, vol. 3, no. 4, pp. 279–294, 2017.

[34] X. Zhang, H. Liu, T. Zhang, Q. Wang, Y. Wang, and L. Tu, "Terminal crossover and steering-based particle swarm optimization algorithm with disturbance," *Applied Soft Computing*, vol. 85, p. 105841, 2019.

[35] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[36] D. A. Van Veldhuizen, *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. Air Force Institute of Technology, 1999.