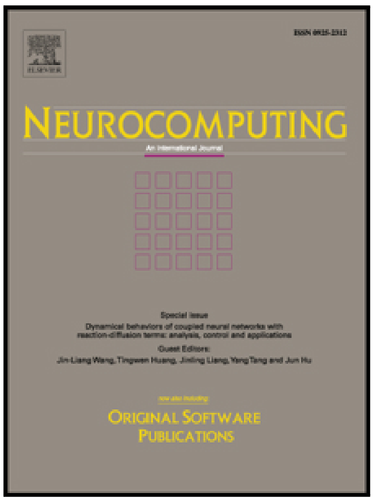
Journal Pre-proof



An improved discrete backtracking searching algorithm for fuzzy multiproduct multistage scheduling problem

Xueli Yan , Yuxin Han , Xingsheng Gu

PII: S0925-2312(20)30256-3
DOI: https://doi.org/10.1016/j.neucom.2020.02.066
Reference: NEUCOM 21934

To appear in: *Neurocomputing*

Received date: 12 August 2019
Revised date: 10 February 2020
Accepted date: 15 February 2020

Please cite this article as: Xueli Yan , Yuxin Han , Xingsheng Gu , An improved discrete backtracking searching algorithm for fuzzy multiproduct multistage scheduling problem, *Neurocomputing* (2020), doi: https://doi.org/10.1016/j.neucom.2020.02.066

This is a PDF file of an article that has undergone enhancements after acceptance, such as the addition of a cover page and metadata, and formatting for readability, but it is not yet the definitive version of record. This version will undergo additional copyediting, typesetting and review before it is published in its final form, but we are providing this version to give early visibility of the article. Please note that, during the production process, errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

© 2020 Published by Elsevier B.V.

Journal Pre-proof

Highlights:

- Set up the recurrence model of fuzzy multistage multiproduct scheduling problem.
- Propose new mutation operations for discrete backtracking searching algorithm.
- Incorporate the information of Gbest into DBSA to accelerate the convergence of it.
- Propose local search related to FMMSP to enhance the exploitation ability of DBSA.
- Design and test the influence of the key parameter on the proposed algorithm.

Journal Pre-proof

*Corresponding author: Xingsheng Gu. TEL.+86-021-6425-3463. Email: xsgu@ecust.edu.cn

Journal Pre-proof

An improved discrete backtracking searching algorithm for fuzzy multiproduct multistage scheduling problem

Xueli Yan[1], Yuxin Han[1], Xingsheng Gu[1]*

([1]Key Laboratory of Advanced Control and Optimization for Chemical Processes (Ministry of Education), East China University of Science and Technology, Shanghai 200237, China)

CRediT author statement

Xueli Yan: Software, Methodology, Validation, Formal analysis, Investigation, Data Curation, Writing - Original Draft, Writing - Review & Editing.

Yuxin Han: Data Curation, Visualization.

Xingsheng Gu: Conceptualization, Resources, Supervision, Project administration, Funding acquisition.

Journal Pre-proof

Abstract: In this study, we address the fuzzy multiproduct multistage scheduling problem (FMMSP). The FMMSP with the objective of minimizing makespan is close to practical manufacturing circumstances in terms of the uncertainty in processing

Journal Pre-proof

time. Backtracking searching algorithm (BSA) is a new proposed meta-algorithm for continuous optimization problems. To solve the FMMSP, an improved discrete BSA, called discrete BSA with local search (DBSA-LS), is developed. The information of the global best solution is incorporated into the mutation process to improve the convergence speed of discrete BSA. A local search related to the FMMSP enhances the exploitation ability of the improved discrete BSA. Left shift operation is applied to improve the solution quality when a hybrid encoding string is decoded into the scheduling scheme. The influence of the key parameter on the performance of the proposed algorithm is tested using different size instances. Sixteen different scales of instances are used to evaluate the performance of the proposed DBSA-LS. Several comparisons are conducted between DBSA-LS and three other algorithms. The results show the effectiveness of the proposed DBSA-LS.

Key words: multiproduct multistage scheduling problem; fuzzy processing time; discrete backtracking searching algorithm; global best solution

Journal Pre-proof

1. Introduction

The increasingly short lifecycles of products increase the requirement to produce high-value-added and low-volume products increases because of the shorter lifecycle of product. Multiproduct or multipurpose batch processes have received increasing attention from researchers and producers because they are suitable for the aforementioned products. The multiproduct batch process, which is common in the production of foods, pharmaceuticals, paints, and specialty chemicals, demands that

Journal Pre-proof

customer orders be processed by the same stage flows. In each stage, several unrelated parallel machines are available and have different processing times for each order. When the material processing of the order is completed in the current stage, the whole batch is preserved and transferred to another unit in the next stage. Thus, batch mixing and splitting, as well as material recycling, are not allowed in a multistage multiproduct batch plant [1].

Multistage multiproduct scheduling of operations is an essential and routine activity to enhance productivity and operability in a multiproduct batch plant. Mathematical programming, which is a linear programming (LP) method, is generally used to solve multiproduct multistage scheduling problem (MMSP). The proposed models are based on two different time representations: discrete- and continuous-time representations. In the discrete-time representation models [2-5], the time horizon is discretized into several time intervals of equal duration. Moreover, the boundaries of these time intervals are the only distribution locations of the events, such as the beginning or end of a task. The discrete-time approach provides a reference grid of time for all operations competing for shared resources. Thus, this approach allows for a straightforward and simple formulation of various constraints, such as inventory costs, and multiple delivery dates. However, the approximation of the time horizon of discrete-time models results in loss of accurate and poor model performance with an unnecessary increase in the number of binary variables.

For the continuous-time representation models, the tasks can occur anywhere on the scheduling horizon. Two different continuous-time models have been reported

Journal Pre-proof

Journal Pre-proof

in the literature: precedence- and time slot-based continuous-time models. Formulations based on precedence are used to model MMSPs. In the precedence-based models [6-8], multiple binary variables for different decisions in the model are restricted by logical relationships. The approach based on precedence expresses the scheduling problem intuitively, but the size of the model grows rapidly with the increase in the number of batches/products. In the time slot-based continuous-time models [9-11], the time slot is defined as the time interval in each unit with unknown duration. Each batch/order must be assigned to the time slot of a unit. To avoid excluding the feasible solutions, sufficient time slots must be postulated, especially for large-size scheduling problems. This consideration can result in empty time slots without any batches/orders assigned to them and the affect overall size of the model.

The approach based on mathematical programming provides a general framework for batch scheduling problems. However, the discrete binary decisions involved, which are inherently combinatorial, in the MMSP model are challenging because of their computational complexity. Modest growth in customer demand can lead to the exponential growth of the problem size and can thus result in a heavy computational burden. Accordingly, the method has difficulty in solving large-size MMSPs.

Some researchers have combined the strength of constraint programming (CP) strategies and LP for MMSPs to reduce the effects of potential combinatorial explosions [1,12]. An important advantage of CP is that it expresses the models in a

Journal Pre-proof

Journal Pre-proof

compact form with procedural and implicitly high-level constructs known as global constraints, revealing problem substructure to the solver that can be missed by other methods. Thus, it can find and obtain satisfactory optimal/suboptimal solutions to the problem rapidly. The LP technique can provide various frameworks for the problem and prove optimality using LP relaxations, while the performance of the hybrid model can degrade substantially and is affected by slight model disturbances for large-size MMSPs.

In contrast to programming strategies, metaheuristic algorithms are stochastic search strategies that perform random steps in the space of feasible solutions. New, better solutions found by the algorithms replace the worst ones with the evaluation of fitness function after every random step. Finally, the best solution obtained by these algorithms at the last iteration time is reserved as the global optimal solution of the problem. Although the global optimality of solutions obtained by metaheuristic algorithms cannot be guaranteed, the quality of the near optimal solutions obtained within a limited time is always satisfactory provided that the algorithms are well designed. Different metaheuristic algorithms have been used to solve different scheduling problems and have been proved by many researchers to be among the most powerful and practical approaches [13-15]. Some works have applied metaheuristic techniques to MMSPs [16-18]. He [16,17] and Shi [18] both decomposed MMSPs into two subproblems of unit assignment and order sequence. Similar rule bases were designed for the unit selection subproblem. He [16,17] employed active scheduling technique, while Shi applied the first-come-first-serve

Journal Pre-proof

Journal Pre-proof

rule to determine the order processing sequence. Genetic algorithms and line-up metaheuristic algorithms were used to solve the MMSP by the two authors, respectively. The results that they obtained demonstrate that the metaheuristic algorithm is more stable and effective than LP in solving large-size MMSPs with different objectives.

The works on MMSPs have been concerned with deterministic processing. An actual manufacturing system encounters uncertainty factors, such as s equipment availability, processing times and costs. Thus, accurate information about some parameters is difficult to obtain in advance. Therefore, these factors in the aforementioned situations should be considered when scheduling problems based on actual production are solved.

Since Bellman and Zedeh proposed fuzzy decision theory in 1970 [19], fuzzy mathematics programming has been an attractive research topic. This development has made fuzzy scheduling a significant topic in scheduling research. However, to the best of our knowledge, no studies have been conducted on the fuzzy MMSP (FMMSP). In the current study, we focus on the FMMSP with the makespan minimization criterion. The order processing time is fuzzy and represented by the triangular fuzzy number (TFN).

The backtracking searching algorithm (BSA), which was originally proposed for the numerical optimization problem, is a novel population-based evolutionary algorithm (EA) proposed by Civicioglu [20]. BSA remembers the historical population information and adds it to the evolutionary process of population. The

Journal Pre-proof

Journal Pre-proof

unique memory mechanism provides BSA with good exploration ability. In addition, BSA has few parameters to control. In recent years, several studies have applied the BSA approach to different discrete optimization problems, such as the community detection problem in complex networks [21], the stochastic dynamic facility layout problem [22], and scheduling problems [23-25]. BSA approaches differ in their means of mapping individuals to a discrete solution. BSA is designed as a hyper-heuristic algorithm to manipulate the sequence of heuristic rules in the low lever for the distributed assembly flow-shop scheduling problem in [24,25]. Each real-value number of individuals in continuous BSA is rounded to an inter-number for which the individuals are decoded into a feasible solution of the combinatorial optimization problem. In [21-23], all BSAs are characterized by inter-number solution representations and discrete operations are proposed to solve discrete optimization problems directly. Insert, swap and inverse operators are usually employed in the mutation process to exchange information between the historical population and the population directly for a discrete BSA. Segment crossover operations, which are similar to the original BSA, are applied to generate new individuals [22,23]. Zou [21] generated a sigmoid function using the XOR operator in the mutation step of BSA. The binary vector manipulates the discrete crossover operation, which is mapped from the historical population individual, the best individual and a random individual from the individual’s neighbors in the evolution population. The author fully used the local information of each individual in the whole process of the proposed discrete BSA. However, all of the BSA variants mentioned above offered no guidance of the current

Journal Pre-proof

Journal Pre-proof

best individual during the evolution process, resulting in a slow convergence speed [26]. Thus, this study developed a discrete BSA suited for the FMMSP with the guidance of the global best (Gbest) individual in the evolutionary population to improve the convergence performance of the proposed algorithm. A local search algorithm related to the FMMSP is also used to enhance the exploitation abilities of the improved algorithm.

The remainder of the paper is organized as follows. Section 2 introduces the mixed integer linear programming (MILP) model of the FMMSP. The recurrence formula of the model is presented after the fuzzy operation is defined. Section 3 introduces the basic BSA. Section 4 describes the discrete BSA in detail and presents some improvements to the discrete BSA. Section 5 first analyses the influence of the key parameter on the improved algorithm and then displays the experimental results of comparisons between the improved BSA and several other algorithms. Section 6 elaborates on the conclusions.

Journal Pre-proof

2. Problem statement

2.1 Fuzzy operators

In the FMMSP model, the processing time of every order per unit is fuzzy and represented as TFN, which has a convex and linear membership function. Several fuzzy number operators are defined, including the addition operator, approximate maximum operation, and ranking operation, to generate a feasible solution to the MMSP.

(1) Addition operator $(+)$

Journal Pre-proof

We assume two fuzzy numbers $\tilde{A} = (a^1, a^2, a^3)$ and $\tilde{B} = (b^1, b^2, b^3)$. Thus, their addition operator is defined by:

$$\tilde{A} + \tilde{B} = (a^1 + b^1, a^2 + b^2, a^3 + b^3). \tag{1}$$

(2) Approximate maximum operation$(\vee)$

The approximate maximum value of two TFNs is determined by the following rules:

$$if \tilde{A} > \tilde{B}, \tilde{I} = \tilde{A}; else \tilde{I} = \tilde{B}. \tag{2}$$

(3) Ranking operation$(> or <)$

The following rules are adapted to rank two fuzzy numbers$\tilde{A} = (a^1, a^2, a^3)$and $\tilde{B} = (b^1, b^2, b^3)$:

$$\text{If } \left(\frac{a^1+2a^2+a^3}{4}\right) > \left(\frac{b^1+2b^2+b^3}{4}\right), \text{then} \tilde{A} > \tilde{B}.$$

$$\text{If } \left(\frac{a^1+2a^2+a^3}{4}\right) = \left(\frac{b^1+2b^2+b^3}{4}\right), a^2 > b^2, \text{then} \tilde{A} > \tilde{B}. \tag{3}$$

$$\text{If } \left(\frac{a^1+2a^2+a^3}{4}\right) = \left(\frac{b^1+2b^2+b^3}{4}\right), a^2 = b^2, a^3 - a^1 > b^3 - b^1, \text{then} \tilde{A} > \tilde{B}.$$

2.2 FMMSP model

In the FMMSP, N customer orders $I = \{1, 2, ..., i..., N\}$ are demanded to be processed in L stages $S = \{1, 2, ..., s, ..., L\}$. Every stage s has n non-identical parallel processing units $U_s = \{1, 2, ..., n\}$ available for an order. Each order is processed only once by exactly one unit per stage. Thus, it must undergo and cannot be processed in more than one unit at the same time. Each unit can handle not more than one order at a time. The processing of the order does not allow for preemption.

Journal Pre-proof

All of the storage tanks used have unlimited storage capacity. The setup times of units and the changeover times between two different orders processed in the same unit are negligible (Fig. 1) [27]. The processing time of order i in unit u is fuzzy and can be represented as $\tilde{T}_{i,u} = \left(T_{i,u}^1, T_{i,u}^2, T_{i,u}^3\right)$. Thus, the fuzzy starting time $\widetilde{Ts}_{i,s}$ and complete time $\widetilde{Tf}_{i,s}$ for order i processed in stage s are fuzzy.

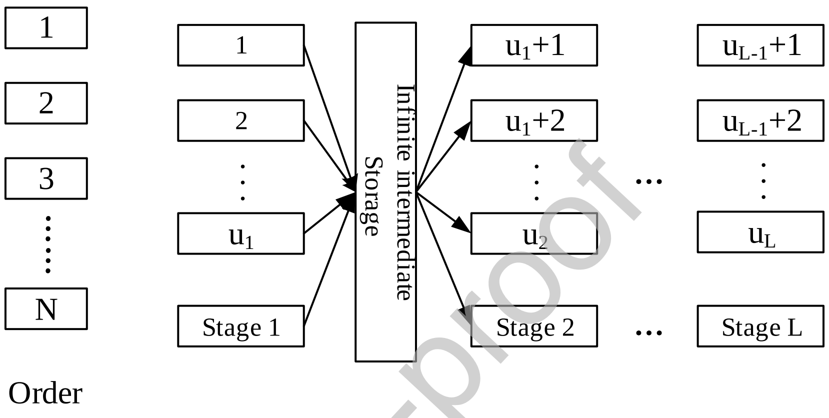


Fig. 1. Multistage multiproduct batch production scheme.

On the basis of the determined formulation of MMSP [28], the FMMSP scheduling model is represented as follows:

$$\min \max(\widetilde{Tf}_{i,L})\ \forall i \in I \tag{4}$$

$$\sum_{u \in U_s} Z_{i,u,s} = 1 \qquad \forall i \in I, s \in S \tag{5}$$

$$\widetilde{Tf}_{i,s} = \widetilde{Ts}_{i,s} + \textstyle\sum_{u \in U_s}\left(Z_{i,u,s}\tilde{T}_{i,u}\right) \forall i \in I, s \in S \tag{6}$$

$$\widetilde{Tf}_{i,s} + M\left(2 - Z_{i,u,s} - Z_{j,u,s}\right) \le \widetilde{Ts}_{j,s}$$

$$\forall i \in I, j \in I, s \in S, u \in U_s, i < j \tag{7}$$

$$\widetilde{Tf}_{i,s} \le \widetilde{Ts}_{i,s+1} \forall i \in I, s \in S \tag{8}$$

Eq. (4) defines that the objective function is to minimize the makespan objective $\tilde{C}$. Eq. (5) states that an order can only be assigned to one unit in every stage. $Z_{i,u,s}$ is a binary variable. When $Z_{i,u,s} = 1$, it represents order i is assigned in unit u in stage s.

Journal Pre-proof

Eq. (6) defines the fuzzy completion time of each order in each stage. The fuzzy start time of order j is expressed as Eq. (7) if order j is processed after order i in the same unit. Eq. (8) enforces that the start of order processing in the next stage must be later than the completion of it in the current stage.

The fuzzy processing time is represented as the TFN, and the addition and approximate maximum operations of the fuzzy number are decomposable. Thus, the model in Eqs. (4) - (8) can be translated into the recurrence form in Eqs. (9) - (13), respectively.

In Eqs. (9) - (13), the auxiliary $\widetilde{TM}_{i,u}$ is the fuzzy completion time of order i in unit u. The fuzzy makespan of FMMSP can be obtained by the linear superposition represented in Eqs. (9) - (13). The approximate maximum operation is used to compute the fuzzy starting time of order $i(i>1)$ in any unit of stage $s(s>1)$. The addition operator is used to obtain the fuzzy completion time of order i in any stage.

Journal Pre-proof

If $s=1, i=1$, then

$$\begin{cases} \widetilde{Ts}_{i,s} = 0 \\ \widetilde{Tf}_{i,s} = \widetilde{Ts}_{i,s} + \tilde{T}_{i,u} \; u \in U_s \\ \widetilde{TM}_{i,u} = \widetilde{Tf}_{i,s}. \end{cases} \tag{9}$$

If $s=1, i>1$, then

$$\begin{cases} \widetilde{Ts}_{i,s} = \widetilde{TM}_{i-1,u} \\ \widetilde{Tf}_{i,s} = \widetilde{Ts}_{i,s} + \tilde{T}_{i,u} \; u \in U_s \\ \widetilde{TM}_{i,u} = \widetilde{Tf}_{i,s}. \end{cases} \tag{10}$$

If $s>1, i=1$, then

$$\begin{cases} \widetilde{Ts}_{i,s} = \widetilde{Tf}_{i,s-1} \\ \widetilde{Tf}_{i,s} = \widetilde{Ts}_{i,s} + \tilde{T}_{i,u} \; u \in U_s \\ \widetilde{TM}_{i,u} = \widetilde{Tf}_{i,s}. \end{cases} \tag{11}$$

If $s>1, i>1$, then

Journal Pre-proof

$$\begin{cases} \widetilde{Ts}_{i,s} = \max\left(\widetilde{Tf}_{i,s-1}, \widetilde{TM}_{i-1,u}\right) \\ \widetilde{Tf}_{i,s} = \widetilde{Ts}_{i,s} + \tilde{T}_{i,u} \qquad u \in U_s \\ \widetilde{TM}_{i,u} = \widetilde{Tf}_{i,s}. \end{cases} \tag{12}$$

$$\begin{aligned} min\ (makespan) &= min\ \ max\left(\widetilde{Ts}_{i,L} + \tilde{T}_{i,u}\right) u \in U_L \\ &= min\ \ max\left(\widetilde{Tf}_{i,L}\right) \\ &= min\, \tilde{C} \\ &= min(C^1, C^2, C^3) \end{aligned} \tag{13}$$

In Eq. (13), (C^1, C^2, C^3) are the best, most probable, and worst completion times, respectively. These times are only related to fuzzy processing time $\tilde{T}_{i,u} = \left(T^1_{i,u}, T^2_{i,u}, T^3_{i,u}\right)$. Eq.13 indicates that the FMMSP is a multiobjective optimization problem. With the ranking rules, the multiobjective optimization problem is transformed into a single-objective optimization problem.

3. Basic BSA

BSA, which is a population-based EA, generates near-optimal solutions of problems using five steps: initialization, selection-I, mutation, crossover, and selection-II [20].

Step 1. Initialization

At the initialization stage of BSA, PS individuals have $D, (D = N * L)$ dimensions. The individuals are generated randomly, subject to uniform distribution in the population of BSA. The initialization step is shown as Eq. (14):

$$P_{i,j} \sim U\left(low_j, up_j\right), i \in N, j \in D\,. \tag{14}$$

Step 2. Selection-I

BSA determines the old population, which is initialized randomly using Eq. (15) before iteration in selection-I. The old population is redefined at the beginning of each iteration using the “if-then” rule shown in Eq. (16) and the permuting function in Eq.

Journal Pre-proof

Journal Pre-proof

(17). Following Eq. (16), the old population for calculating the search direction retains the historical experience of the population, which belongs to a randomly selected previous generation until it is updated. In accordance with this description, BSA memorizes the historical information of the population randomly and constructs the search direction matrix with it. This mechanism distinguishes it from the other EAs and increases the information diversity of the population, providing BSA with good exploration ability.

$$oldP_{i,j} \sim U\left(low_j, up_j\right) \tag{15}$$

$$if\ a < b, then\ oldP := P \,|\, \left(a, b\right) \sim \left(0,1\right) \tag{16}$$

$$oldP := permuting\left(oldP\right) \tag{17}$$

Step 3. Mutation

In the mutation stage, a trial population is generated using Eq. (18), in which the amplitude of the search-direction matrix $oldP - P$ is controlled by parameter F .

$$M\,u\,t\,a\,n{=}t \quad +P.\left(\ F \quad o{+}d\right) \tag{18}$$

Step 4. Crossover

BSA has more complex crossover operators than other EAs. The crossover stage of BSA includes two step operators. In the first step, a binary integer-valued matrix (map) of size is generated by Eqs. (19) and (20). In Eq. (19), the mix rate parameter controls the number of elements that will crossover. Once the map is obtained, the trial population is updated using Eq. (21). After the main crossover steps are completed, BSA checks whether the new population individual is in the defined feasible domain; if not, then the boundary control mechanism is triggered, as shown

Journal Pre-proof

in Eq. (22).

$$Map_{i,u(1:[mixrate*rand*D])} = 1 \mid u = permuting(1,2,3...,D) \quad (19)$$

$$Map_{i,randi(D)} = 1 \quad (20)$$

$$Crossover = P + (map.*F).*(oldP - P) \quad (21)$$

$$Crossover_{i,j} = rnd.(up_j - low_j) + low_j, if\ Crossover_{i,j} < low_j\ or\ Crossover_{i,j} > up_j \quad (22)$$

Step 5. Selection-II

When the new trial population is obtained, BSA compares the fitness of the new individuals with that of the original ones in population P and the current optimal individual. If the new one is better than the individuals in P, then it replaces the original individual. Meanwhile, if it is better than the optimal individual, then it replaces the best ones.

The basic BSA, based on the addition and subtraction of real numbers, is not adapted for the FMMSP. To solve the FMMSP efficiently, a discrete BSA with a local search (DBSA-LS) is proposed and discussed in the next section.

Journal Pre-proof

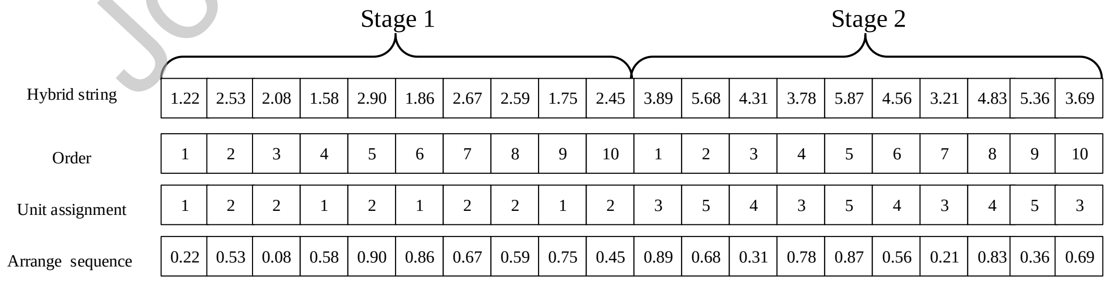
4. DBSA-LS for FMMSP

BSA is an algorithm which is adapted to the continuous optimization problem. Thus, it cannot be applied directly to solve the FMMSP, which is a combinatorial optimization problem. A discrete BSA is proposed and presented in this section. The Gbest solution of the population is contained to improve the convergence speed of the discrete BSA. Left shift operation eliminates the wasting of idle time for the orders in the units. Consequently, the solution quality of scheduling is improved. The local search based on the insert operation is implemented to enhance the exploitation ability of the proposed algorithm. With these technologies, the proposed algorithm, called DBSA-LS, is expected to provide high-quality solutions for the FMMSP.

4.1 Encoding and initialization

The solution of the problem with appropriate encoding should be provided to use BSA in solving the FMMSP. In the FMMSP, two problems, including the unit selection per stage and the order sequence in each unit, are solved. Thus, a hybrid string with integer and random keys is used to represent the solution to the FMMSP. For an FMMSP with *N* orders and *L* stages, the string of solutions has *L* sections and *N* numbers per section. The size of the string is *N*×*L*. Every number per section consists of integer and decimal parts. The integer part is the number of the assigned unit for an order and the sequence of the decimal part determines the arrangement sequence of orders in a unit. Fig. 2 illustrates a sample string structure for a 10×2 FMMSP. Two and three units are available for every order in the first and second stages, respectively.

Journal Pre-proof



	Stage 1										Stage 2									
Hybrid string	1.22	2.53	2.08	1.58	2.90	1.86	2.67	2.59	1.75	2.45	3.89	5.68	4.31	3.78	5.87	4.56	3.21	4.83	5.36	3.69
Order	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
Unit assignment	1	2	2	1	2	1	2	2	1	2	3	5	4	3	5	4	3	4	5	3
Arrange sequence	0.22	0.53	0.08	0.58	0.90	0.86	0.67	0.59	0.75	0.45	0.89	0.68	0.31	0.78	0.87	0.56	0.21	0.83	0.36	0.69

Fig. 2. Sample hybrid encoding string.

As shown in Fig. 2, the hybrid string with the same integer part indicates that the orders are assigned to the same unit. For example, in stage 1, orders $\{1,4,6,9\}$ are

Journal Pre-proof

processed in unit 1, and their random key values are $\{0.22, 0.58, 0.86, 0.75\}$. By sorting random key values in ascending order, their arrangement sequence is $\{1, 4, 9, 6\}$ in unit 1. Orders $\{2, 3, 5, 7, 8, 10\}$ are processed in unit 2, and their arrangement sequence is $\{3,10,2,8,7,5\}$.With the same operation, the unit allocation and arrangement sequence per stage of each order can be determined. The unit number differs between different stages for each order. With hybrid encoding, the information exchange between different stages would introduce the infeasible solutions. For example, if unit 3 appears in any location of stage 1, then the solution is infeasible. The operation of information exchange between different stages will not be implemented to avoid infeasible solutions of the problem.

Before every metaheuristic algorithm performs the steps for optimization, an initialization population with a certain size must be generated as the starting point of the optimization process. This process is the same for DBSA-LS. In the initialization of DBSA-LS, we generate all of the *PS* individuals at random.

4.2 Left shift operation

The left shift operation, which is similar to the active scheduling technique discussed in [16], is implemented in the DBSA-LS algorithm to improve the fuzzy solution quality. In the determined MMSP, idle time could be observed because of the processing time constraints between different stages. The starting time of an order must be later than its completion time in the previous stage. If an order that is completed later than other orders in the previous stage has a front processing position in the current stage, then the unit could have considerable idle waiting time before it

Journal Pre-proof

Journal Pre-proof

starts any operation. For example, the last order completed in the previous stage is in the first processing position of a unit in the current stage. Without left shift operations, the solution quality will be poor.

Table 1. Processing data of example 1.

Order	Unit 1	Unit 2	Unit 3
	Stage 1	Stage 2	
1	(4,5,6)	(5,6,7)	(3,4,6)
2	(7,8,10)	(12,13,14)	(10,11,12)
3	(10,11,12)	(4,5,6)	(6,8,10)
4	(15,17,20)	(12,14,15)	(10,12,13)

In the FMMSP, fuzzy idle time might appear because of the same reason in the determined MMSP. Table 1 provides the processing data of example 1 with four orders and three units for two stages. The Gantt chart of scheduling for a feasible solution of the FMMSP {1.23,1.54,1.46,1.70,2.80,2.73,3.12,2.58} without the left shift operation is shown in Fig. 3. In the figure, the TFN below the vertical coordinates represents the fuzzy starting time of orders in the unit and that above the vertical coordinates is the fuzzy completion time of orders in the unit. TFN of the same order has the same color. The minimum fuzzy completion time of example 1 is (65,74,84) for the feasible solution without the left shift operation.

The left shift operation for the FMMSP is used to reduce unnecessary, idle time in scheduling. For order i in stage s, its processing operation in the current stage can be started unless the operation in the previous stage is completed, and the unit that it is assigned to is idle. The fuzzy starting time of order i is shown in Eq. (23):

$$\widetilde{Ts}_{i,s} = max(\widetilde{TM}_{i-1,u}, \widetilde{Tf}_{i,s-1}). \quad (23)$$

Journal Pre-proof

Journal Pre-proof

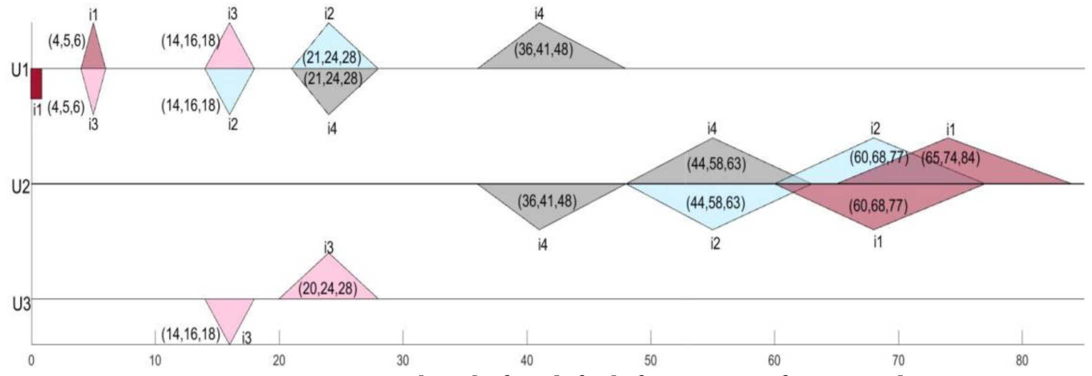


Fig. 3. Fuzzy Gantt chart before left shift operation for example 1.

The fuzzy completion time $\widetilde{Tf}_{i,s}$ $(s = 0)$ of every order is $(0,0,0)$. A dummy order $i = 0$ is assigned to the first location in each unit, and the fuzzy starting time and completion times of this order are $(0,0,0)$. When an order $i(i > 0)$ is to be assigned to unit $u(u \in s, s > 1)$, all of the idle time intervals from left to right are checked for whether they are sufficiently long for the processing of order $i(i > 0)$ without any change to the assigned order. We suppose that the fuzzy intervals are represented as $[\tilde{T}_k^s, \tilde{T}_k^f]$. Thus, order $i(i > 0)$ is arranged in the first intervals to satisfy Eq. (24):

$$max(\tilde{T}_k^s, \tilde{T}_{i,s-1}^f) + \tilde{T}_{i,u} \le \tilde{T}_k^f, u \in s. \tag{24}$$

As shown in Fig. 4, for unit 2, the fuzzy idle time (32,36,42) is sufficient to process order $i1$ with the fuzzy processing time (5,6,7) according to Eqs. (23) and (24). After order $i1$ has been shifted to the left, the new fuzzy idle intervals (15,17,20) between order $i1$ and order $i4$ are sufficient for processing order $i2$ with the fuzzy processing time (12, 13, 14). After the left shift operation, the minimum maximum fuzzy completion time of the example is decreased to (48,55,63). Evidently, the quality of a feasible solution is enhanced.

Journal Pre-proof

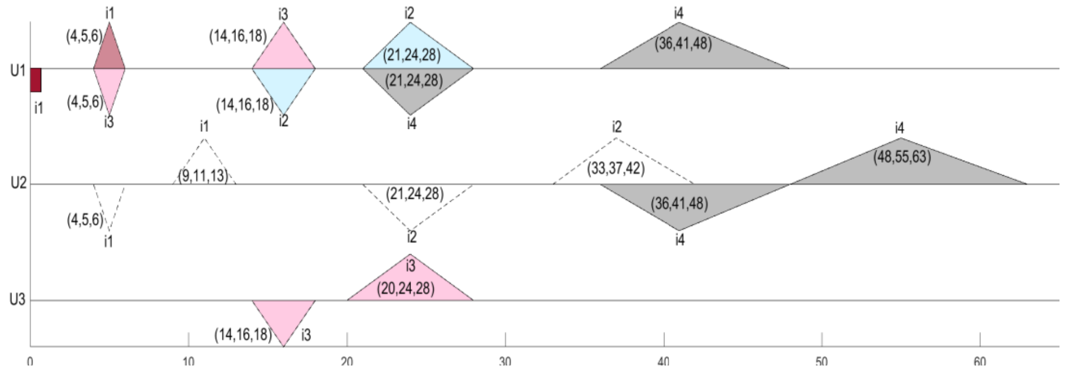


Fig. 4. Fuzzy Gantt chart after shift operation for example 1.

4.3 Selection-I

The selection of the old population enables BSA to remember population historical information randomly. This feature enlarges the information source of the population and is the core idea of BSA. However, if the historical information in the current old population is helpless and is not updated for many iteration times, the evolution of the population will stagnate. Thus, the old population will be updated for every five iterations.

4.4 Mutation

In DBSA-LS, two new discrete operations “$\otimes$” and “$\odot$” are used to generate the new mutation that is suitable for the FMMSP. The “$\otimes$” operation represents the discrete migration operation and the “$\odot$” operation decides the migrating length F. The best individual of the population has the best fitness value and always carries considerable helpful information about the problem. It guides the population in the right direction rapidly if used appropriately. In the mutation of DBSA, the best individual information is referenced. The discrete mutant operation for DBSA is defined in Eqs. (25) and (26).

Journal Pre-proof

Journal Pre-proof

$$Mutant_i = \begin{cases} (P_i \oplus oldP_i) \odot F \ \ if\ rand < MC \\ (gbest \oplus oldP_i) \odot F\ if\ rand \geq MC \end{cases} \quad (25)$$

$$F = ceil(rand * N) \quad (26)$$

For the discrete scheduling problem, the best individual of the population might not be non-improving for certain iterations. If the information of the best individual is referenced directly, the population would easily be trapped in the local optimal solution. Thus, in Eq. (25), the perturbation from $oldP_i$ is imposed on the best individual by $(gbest \oplus oldP_i)$ before it is referenced as the mutation individual. This operation is designed to restrain the premature convergence of the algorithm. *MC* is the selection probability of two different mutation operations.

In Eq. (25), the migration operator "$\otimes$"migrates some bits from the individual of the old population to the current or best individual of the population with probability MC, which is in the interval [0,1]. If the random number is greater than or equal to *MC*, then the migration operation is from the individual of the old population to the best individual of the population. Otherwise, the migration operation is from the individual of the old population to the current individual of the population. A small value of *MC* indicates a strong influence of the information of the best individual on the new population. When *MC* is set to the minimum value of 0, only the information of the best and historical individuals is referenced. The population has fast convergence speed but is trapped in the local solution easily. When *MC* is set to the maximum value of 1, only the information of the current and historical individuals is referenced, resulting in poor search efficiency of the algorithm. DBSA-LS can perform well only with a good balance of the current, historical, and best individual information.

Journal Pre-proof

Journal Pre-proof

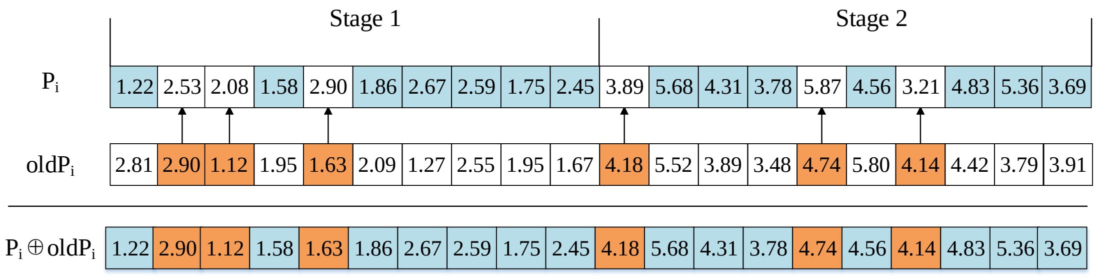


Fig. 5. Migration operation.

The number of migration operations is decided by *F*, which is a rand integer number in the interval. The hybrid encoding string for the FMMSP has L sections, and the number of migration operations is the same for each section. However, the locations of the migration operations are selected randomly in each section. *F*L* bits are changed in the current or best individual for every migration operation. The example in section 4.1 is still cited here. We suppose that the value of *F* is 3, and the random locations per stage are $\{2,3,5;1,5,7\}$. The migration operation is shown in Fig. 5.

4.5 Crossover operations

In discrete BSA, one- and two-segment crossover operations are adopted to generate the trial population. We use the two-segment crossover strategy shown in Fig. 7 to explain the implementation rule in the solution of the FMMSP with multisections. Parents A and B are the hybrid strings to be implemented in the crossover operation. Four positions are randomly generated in the interval $[1,N]$ and are then ranked. We suppose that the ranked position is (2,4,6,8) for the example in section 4.1. The first segment between the second and fourth positions from parent B is transmitted to its child in each stage. The second segment between the sixth and eighth positions from parent B is transmitted to its child in each stage. Then, the child

Journal Pre-proof

values of other positions are obtained from parent A. One child is generated in every crossover operation, and no infeasible solutions exist. Parent A is the mutation individual newly produced, and parent B is the current individual of the population. The one-segment crossover operation is similar to the two-segment crossover operation. Figs. 6 and 7 show the details of the crossover operations for a multisection string.

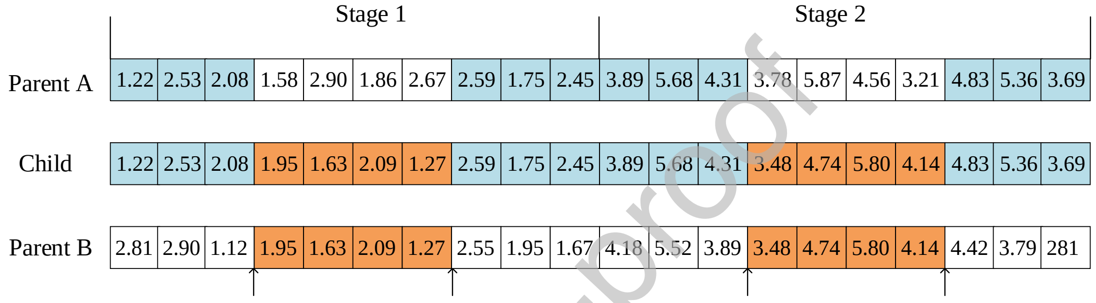


Fig. 6. One-segment crossover operation.

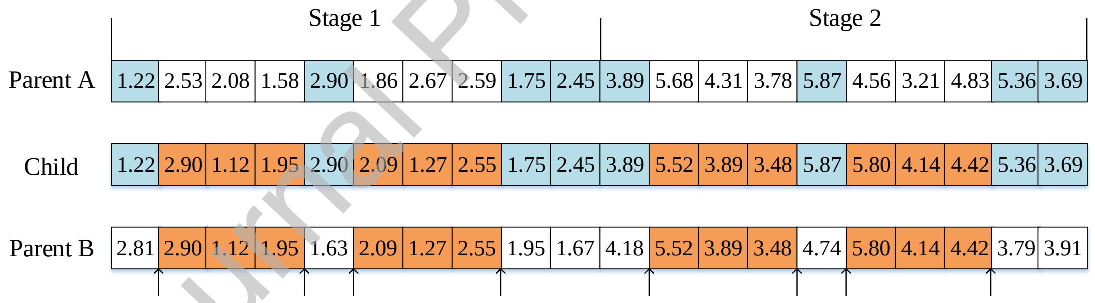


Fig. 7. Two-segment crossover operation.

The fitness of the child individuals is evaluated and compared with that of the original individuals in population P and the optimal individual at present. If the new trial individual is better than the individuals in P, then it replaces the original individual. If it is also better than the best individual, then it replaces the optimal individual.

4.6 Local search

Journal Pre-proof

Journal Pre-proof

Permutation scheduling problems have three common neighborhood structures: insert, swap, and inverse. The neighborhoods are obtained by removing a job and inserting it at another position of the string, exchanging two jobs in any two positions and inverting the sequence between any two positions, respectively. Many researchers have proved that the insert neighborhood structure is better than the two other neighborhoods [29,30]. In this study, the insert neighborhood is used to find a better solution around the optimal solution for improving the exploitation ability of DBSA. Different from scheduling based on permutation, the insert neighborhood of the hybrid string is the neighborhood composition of the machine selection and sequence problems. When an insert operation is applied to a hybrid string, the units assigned to and the arrangement sequence of the orders are both impacted. The left shift operation can compensate the influence for the insert operation at stage $s(s>1)$ because the sequence of order in the unit of stage $s(s>1)$ is decided by the arrangement sequence and the left shift operation. Thus, the insert operation is only applied in the first stage. Fig. 8 shows the insert operation of the hybrid string.

	Stage 1									
Parent	1.22	2.53	2.08	1.58	2.90	1.86	2.67	2.59	1.75	2.45
Child	1.22	2.53	1.75	2.08	1.58	2.9	1.86	2.67	2.59	2.45

Fig. 8. Insert operation in the first stage.

Before the insert operation is implemented, the orders $\{1,4,6,9\}$ are assigned to unit 1, and the arrangement sequence is $\{1,4,9,6\}$. The orders $\{2,3,5,7,8,10\}$ are assigned to unit 2, and the arrangement sequence is $\{3,10,2,8,7,5\}$. With the insert

Journal Pre-proof

Journal Pre-proof

operation, the orders $\{1,3,5,7\}$ are assigned to unit 1, and the arrangement sequence is $\{1,5,3,7\}$. The orders $\{2,4,6,8,9,10\}$ are assigned to unit 2, and the arrangement sequence is $\{4,10,2,9,8,6\}$. The insert operation changes the unit assignment and the order sequence but retains the order number for each unit. Evidently, the insert neighborhood of the FMMSP is more complicated than that of the scheduling based on permutation.

Fig. 9 shows the pseudo code of the local search based on the insert operation. In Fig. 9, ω and υ are two different integer numbers in the interval $[1, N]$. φ is the hybrid string, and φ_{best} is the optimal solution of the population. The insert $(\varphi, \omega, \upsilon)$ operation represents inserting the value of the υth position into the ωth position in the first stage of φ.

The local search based on the insert operation is used to improve the exploitation ability of DBSA. The objective evaluation for the FMMSP has time complexity. Thus, the local search for the optimal solution is implemented when the optimal solution of the problem searched by DBSA has not been non-improving for *NIP* iterations, and the insert operation is executed *PS*/2 times for the local search. If the new best solution is better than the original optimal solution, then the new best solution replaces it as the global optimal solution of the population.

Journal Pre-proof

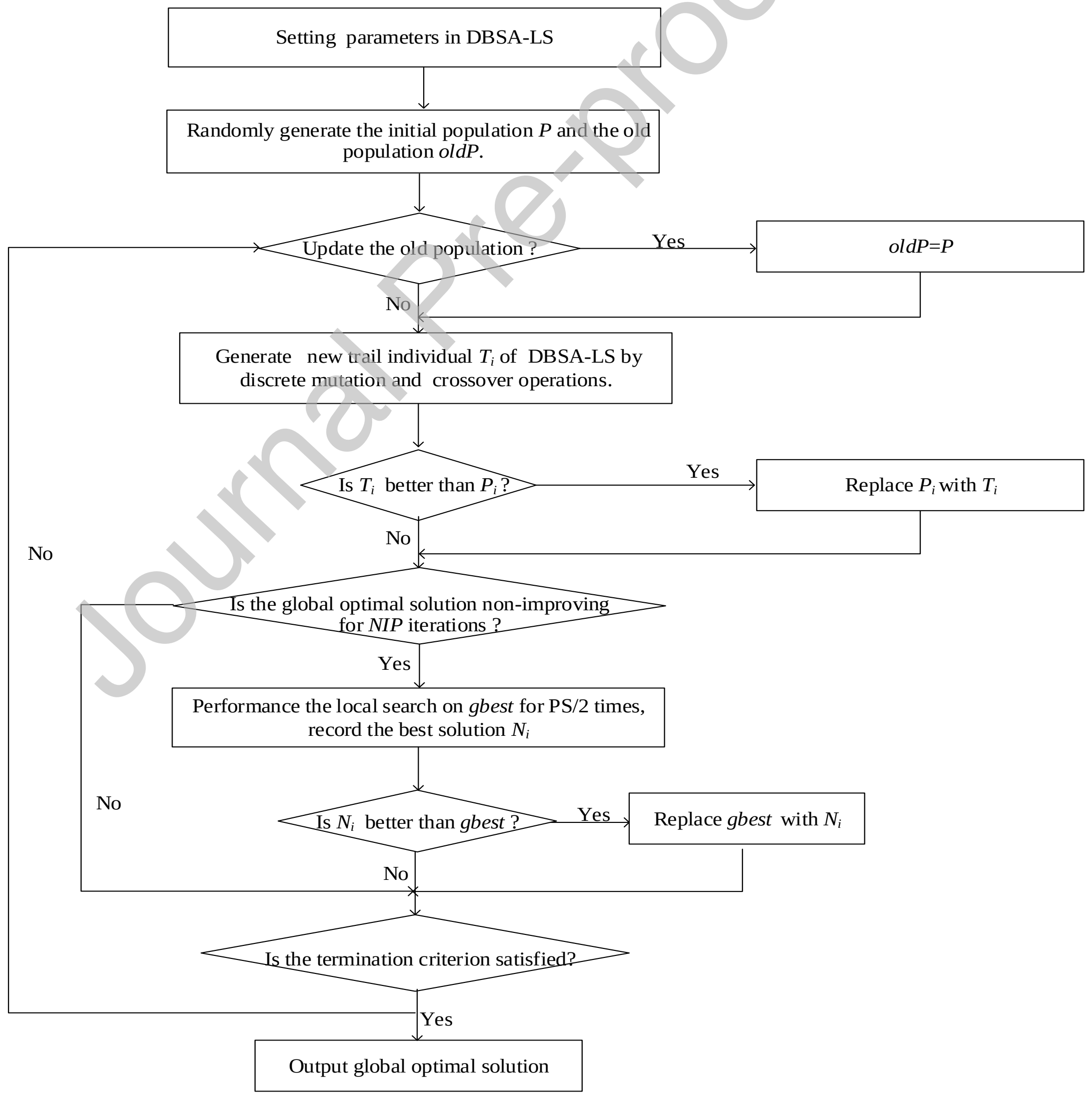
```
$loop$=0
    randomly generate $\omega, \upsilon$ and $\omega \neq \upsilon$
    $\varphi' = \text{insert}(\varphi_{best}, \omega, \upsilon)$
    do{
    randomly generate $\omega, \upsilon$ and $\omega \neq \upsilon$
     $\varphi'' = \text{insert}(\varphi', \omega, \upsilon)$
```

Journal Pre-proof

```
        if $\left(\text{Fitness}\left(\varphi''\right) < \text{Fitness}\left(\varphi'\right)\right)$
        $\varphi' = \varphi''$
        endif
    $loop{+}{+}$
  $\}$ while $\left(loop < \left(PS/2\right)\right)$
        if $\left(\text{Fitness}\left(\varphi'\right) < \text{Fitness}\left(\varphi_{best}\right)\right)$
          $\varphi_{best} = \varphi'$
    Endif
```

Fig. 9. Pseudo-code of insert local search.

Based on the aforementioned descriptions, the flow chart of DBSA-LS and the flow diagram of DBSA-LS are shown in Fig.10 and summarized in Algorithm 1.



Setting parameters in DBSA-LS
Randomly generate the initial population P and the old population oldP.
Update the old population ?
Yes
oldP=P
No
Generate new trail individual T_i of DBSA-LS by discrete mutation and crossover operations.
Is T_i better than P_i ?
Yes
Replace P_i with T_i
No
No
Is the global optimal solution non-improving for NIP iterations ?
Yes
Performance the local search on gbest for PS/2 times, record the best solution N_i
No
Is N_i better than gbest ?
Yes
Replace gbest with N_i
No
Is the termination criterion satisfied?
Yes
Output global optimal solution

Fig.10. The flow chart of DBSA-LS

Journal Pre-proof

Journal Pre-proof

```
Algorithm 1: DBSA-LS
1.Begin
    2.Initialize parameter population size PS, the mutation selection parameter MC
    and the allowed maximum consecutive non-improving iterations (NIP);
    3. Initialize and evaluate the evolution population and the historical population.
    4. While (stopping condition not met)
    5. Check whether the historical population has not been updated for 5 iterations.
    If yes, update it. Then, conduct permutation on the historical population.
    6. Execute the new discrete mutation operation according to Eqs.(25)-(26) and
    the crossover operations to generate the trial individual $T_i$ for each individual $X_i$.
    7. Evaluate every trial individual $T_i$ and accept it if it is better than $X_i$.
    8.Check whether the Gbest solution of the population has been non-improving
    for NIP iterations. If yes, execute the local search algorithm for $PS/2$ times and attain
    the best solution $N_i$.
    9. Evaluate $N_i$ and accept it if it is better than the Gbest solution.
    End while.
End while.
```

5. Computational results and discussion

5.1 Experimental setup

No benchmark sets can be used directly to validate the effectiveness of the proposed DBSA-LS approach in solving the FMMSP because of a lack of research on FMMSP. Thus, the experimental data are generated randomly following some rules. The number of orders in the present study has four configurations, that is, $N = (10,20,30,40)$. The number of stages is four, that is, $L = (2,3,4,5)$. In summary, $N \times L$ combinations are tested. The unit number is randomly uniformly distributed at between two and four per stage. Three characteristics that define a problem are the number of orders, the number of stages and the number of total non-identical units in all stages. The authors employ the notation of o10s2u5, for instance, which means a problem with 10 orders, 2 stages and 5 units. The letter o, s and u represent the orders,

Journal Pre-proof

Journal Pre-proof

stages and units in total, respectively. Sixteen computational experiments are generated based on using the generation procedure for FMMMP in different sizes. Problem o10s2u5 with 10 orders, 2 stages and 5 units are categorized as simple problems. Seven instances are categorized as small-size problems, and they are o10s3u8, o10s4u9, o10s5u14, o20s2u6, o20s3u7, o20s4u11, and o30s2u6. Five instances o20s5u12, o30s3u10, o30s4u11, o40s2u7, o40s3u8 are categorized as medium- size problems, and the large-size problem test subset includes the instances o30s5u12, o40s4u12 and o40s5u12.

A determinable number, which is uniformly distributed in the range [5,40], is generated and preserved as the most possible processing time of the order to obtain a fuzzy processing time for every order in a unit. The shortest processing time is a random number in the range$[\delta_1 t, t], \delta_1 \in [0,1]$. The longest processing time is a random number in the range $[t, t\delta_2], \delta_2 > 1$. In this way, a fuzzy processing time $[\delta_1 t, t, \delta_2 t], \delta_1 \in [0,1], \delta_2 > 1$ is obtained. δ_1 is set at 0.9, and δ_2 is set at 1.2.

The algorithms are programmed in MATLAB software and run on a PC with Intel an Intel(R) core, 3.4 GHZ station, and 4.0 GB of RAM.

5.2 Parameter setting

The parameter setting plays an important role in intelligent optimization algorithms, specifically for DBSA-LS. Three parameters are included in the DBSA-LS, and they are the population size *PS*, the mutation selection parameter *MC* and the allowed maximum consecutive non-improving iterations (*NIP*). To study the impact of the parameters on the performance of the DBSA-LS, taguchi method of

Journal Pre-proof

Journal Pre-proof

design of experiment (DOE) [31] is constructed on five instances, including o10s2u5, o10s3u8, o20s3u7, o30s4u11 and o40s5u12, which are from the four test subsets. The termination condition is set to be the maximum iteration times 2**N***U,* considering the influence of the problem size, which is mainly determined by the total number of orders *N* and the total number of units *U*. Parameter *MC* is set in the range of [0,1]; The parameters *PS* and *NIP* change in accordance with the problems. The parameter *NIP* decides the trigger frequency of the local search. A small *NIP* value can help the algorithm to search the space sufficiently, while it incurs a heavy computational burden, especially for the large-size problems. A large *NIP* value is contrary. Thus, a flexible level design of *NIP* is necessary for different instances. Three reasonable parameter levels for each parameter are determined for five instances and are shown in Table 2. The orthogonal array L9(3^3) is carried out on the experiments to determine the best parameter combination.

Journal Pre-proof

Table 2. Levels of the parameters of DBSA-LS for five instances.

Parameters	Parameter level														
	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
MC	0.1	0.4	0.7	0.1	0.4	0.7	0.1	0.4	0.7	0.1	0.4	0.7	0.1	0.4	0.7
PS	60	100	140	100	140	180	100	140	180	180	220	260	200	240	280
NIP	5	20	35	5	20	35	15	45	75	15	45	75	15	45	75

Considering the experiment on instance o10s3u8 for example, Table 4 shows 9 groups of the orthogonal parameter test samples. For the minimum fuzzy makespan objective, the value $\left(\frac{C^1+2*C^2+C^3}{4}\right)$ is designed to be the response value. For each parameter combination, the DBSA-LS is run 10 times independently. The average values of the response values are obtained and listed in Table 3.

Table 3. Orthogonal parameter table L9 (3^3) and the results of o10s3u8.

No.	Parameter			Average value
	NC	*PS*	*NIP*	
1	1(0.1)	1(100)	3(35)	103.8750
2	2(0.4)	1(100)	1(5)	103.7000
3	3(0.7)	1(100)	2(20)	104.6500
4	1(0.1)	2(140)	2(20)	103.4250
5	2(0.4)	2(140)	3(35)	103.8000
6	3(0.7)	2(140)	1(5)	102.8750
7	1(0.1)	3(180)	1(5)	104.3250
8	2(0.4)	3(180)	2(20)	103.5000
9	3(0.7)	3(180)	3(35)	103.9250

The response values representing the significance rank of the parameters are described in Table 4, indicating that *PS* is the most important parameter, and *NIP* and *MC* rank second and third for the instance o10s3u8, respectively. The parameter level trends are described as in Fig. 11, according to which the parameter values for instance o10s3u8 are set as follows: *MC*=0.4; *PS*=140; and *NIP*=5.

Table 4. The mean response values of DBSA-LS for the instance o10s3u8.

Level	Parameter		
	MC	*PS*	*NIP*
1	103.8750	104.0750	**103.6333**
2	**103.6667**	**103.3667**	103.8583
3	103.8167	103.9167	103.8667
Delta	0.2083	0.7083	0.2334
Rank	3	1	2

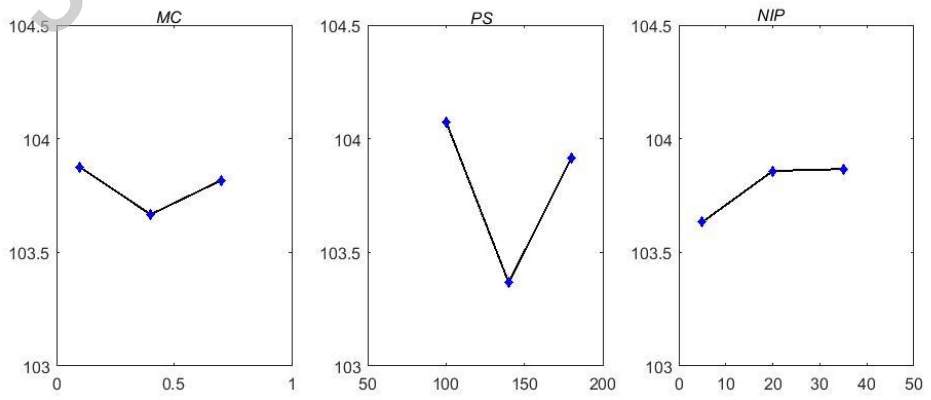


Fig. 11. Factor level trend of the influence of parameters on algorithm performance.

Journal Pre-proof

Journal Pre-proof

Table 5 lists the response values representing the significance ranks of parameters for the remaining four instances.

Table 5. The mean response values of DBSA-LS for different instances.

Problem	Level	Parameter		
		MC	*PS*	*NIP*
O10s2u5	1	45.4000	45.6417	**45.2500**
	2	**45.3167**	**45.2000**	45.3583
	3	45.3583	45.2333	45.4667
	Delta	0.0833	0.4417	0.2167
	Rank	3	1	2
O20s3u7	1	157.4417	157.8917	**156.4917**
	2	**155.8750**	**156.0333**	156.7333
	3	156.9000	156.2917	156.9917
	Delta	1.5667	1.8584	0.5
	R	2	1	3
O30s4u11	1	292.5250	292.7583	291.9083
	2	**290.4667**	**291.7333**	**291.7583**
	3	293.6417	292.1417	292.9667
	Delta	1.8584	1.026	1.2085
	R	1	3	2
O40s5u13	1	382.4667	382.8167	**382.4500**
	2	**382.0333**	**382.8083**	383.3750
	3	384.3500	383.2250	383.0250
	Delta	2.3167	0.4167	0.9250
	R	1	3	2

The mutation selection probability *MC* determines the influence of the best individual on the population. Table 5 indicates that the parameter *MC* plays a critical role when DBSA-LS solves the medium and large-size problems as o30s4u11 and o40s4u13, respectively. When *MC* is set at 0.4, DBSA-LS has a good balance between the convergence speed and precision with the Gbest solution guidance. For all four instances, according to Table 5, the parameter value of *MC* is set at 0.4. Furthermore, the parameter values of *PS* are set as 100, 140, 220, and 240, respectively and *NIP* is set as 5, 5, 45, and 15, respectively.

For the remaining instances in experimental sets, the parameter values of

Journal Pre-proof

Journal Pre-proof

DBSA-LS are set to be the same as those of the tested instances in the same subset because of the similar problem size.

5.3 Comparison with other algorithms

To evaluate the performance of DBSA-LS in solving the FMMSP, it is compared with three other algorithms, namely BDBSA, MDBSA [22], and IGA [32]. Here, the basic discrete BSA (BDBSA) has the same discrete mutation and crossover operations with DBSA-LS. In BDBSA, all individuals are initialized randomly, and no local search operation is applied. Further, the value of *MC* is set at 1, indicating that the evolution of BDBSA has no guidance of the best individual of the population. Additionally, the historical information is updated randomly, which is the same as in the original BSA. BDBSA randomly selects a crossover operation between one- and two-segment crossover operations for evolution. MBSA is a discrete modified BSA, which differs in the mutation and crossover operations from the proposed algorithm. The population size and iteration times of the comparison algorithms are the same as those of DBSA-LS to ensure a fair comparison. The best, worst, and average values for different size problems obtained by DBSA-LS, BDBSA, MDBSA, and IGA algorithms over 10 independent runs are listed in Tables 6 and 7. Table 6 lists the results of the simple and small-size problems, and Table 7 is designed for the results of the medium and large-size problems. To render comparisons clearer, the best of the best, mean and worst values among all of the algorithms appear in boldface.

Journal Pre-proof

Table 6. Computational results of various algorithms for the simple and small-size problem.

Instance	Algorithm	Best value	Mean value	Worst value
o10s2u5	DBSA-LS	**(36,44,52)**	**(38,45, 52)**	(42,48,53)
	BDBSA	**(36,44,52)**	(38.60,45.50,52.00)	**(40,46,54)**

Journal Pre-proof

	MBSA	(40,47,55)	(44,50,57)	(42.2,49.3,54.6)
	IGA	(39,45,52)	(40.10,46.90, 54.50)	(43, 49,57)
o10s3u8	**DBSA-LS**	**(89,99, 115)**	**(93.60,101.90, 114.10)**	**(99, 105,110)**
	BDBSA	(95,103 ,111)	(96.70,105.30, 112.10)	(99, 106,117)
	MBSA	(94,103, 118)	(97.60, 106.10,119.00)	(100,109,120)
	IGA	(91,99, 115)	(95.15,104.00,117.35)	(98,106,120)
o10s4u9	**DBSA-LS**	**(113, 127, 130)**	**(119.90, 132.60, 137.10)**	**(126, 141, 145)**
	BDBSA	(123,135,140)	(126.90,140.30, 143.70)	(129, 144, 149)
	MBSA	(131,145, 149)	(139.00,151.00,152.00)	(133.9,148.2,152.0)
	IGA	(119,130,134)	(127.60,141.30,147.60)	(138, 152, 158)
o10s5u14	**DBSA-LS**	**(119, 131,135)**	**(122.10,133.00,137.50)**	**(128, 136, 150)**
	BDBSA	(121,136,137)	(126.00,138.00,141.70)	(128, 141, 145)
	MBSA	(128,138,156)	(131.10,143.50,152.00)	(132,146,159)
	IGA	(120,131,135)	(125.10,137.70,143.60)	(134,144,145)
o20s2u6	**DBSA-LS**	**(109, 118,119)**	**(110.55,121.90,123.85)**	**(113,127,128)**
	BDBSA	(111, 122,125)	(113.20,125.20,130.80)	(118,126,130)
	MBSA	(109,120,137)	(116.54,127.97,134.76)	(121,132,133)
	IGA	(109.5,119,123.5)	(112.65,122.80,126.85)	(118.5,128.0,126.5)
o20s3u7	**DBSA-LS**	**(129, 146,167)**	**(139.50, 157.50,174.20)**	**(140, 160, 186)**
	BDBSA	(144, 159,163)	(142.65,159.50,177.15)	(147,164,179)
	MBSA	(153,169,172)	(152.75,170.30,179.55)	(157,174,177)
	IGA	(138, 155,176)	(152.00,168.00,195.00)	(143.8,161.7,180.5)
o20s4u11	**DBSA-LS**	**(181,204, 229)**	**(190.90,208.80,236.20)**	**(199,215,249)**
	BDBSA	(190,206,235)	(389.20,428.00,466.10)	(195,213,245)
	MBSA	(199, 218,253)	(205.30,225.10, 244.30)	(205,226,259)
	IGA	(189, 206,230)	(195.25, 212.60,237.45)	(214,222,236)
o30s2u6	**DBSA-LS**	**(208,227,259)**	**(211.10,231.00, 259.70)**	**(213,233,266)**
	BDBSA	(212,234,266)	(214.65,236.30,267.10)	(216,237,273)
	MBSA	(212, 233,264)	(211.80, 234.60,265.00)	(211, 235, 268)
	IGA	(208, 230,262)	(210.60, 232.50,264.40)	(215,235,268)

Table 6 shows that DBSA-LS and BDBSA yield better solutions than MDBSA and IGA for the simple problem o10s2u5. Even with restarting operations, the solutions of the IGA severely converge when it solves the simple problem using the same population size as DBSA-LS. For the small, medium and large-size problems, DBSA-LS outperforms three other comparison algorithms. From Tables 6 and 7, it is notable that the exploitation ability of BDBSA degrades for more complex FMMSPs, such as small, medium and large-size problems, compared with DBSA-LS and IGA. Thus, it illustrates that the proposed measures to improve the exploitation ability of DBSA-LS are critical and effective for the FMMSP. MBSA has the worst performance for all of the size problems among the four algorithms. We can conclude

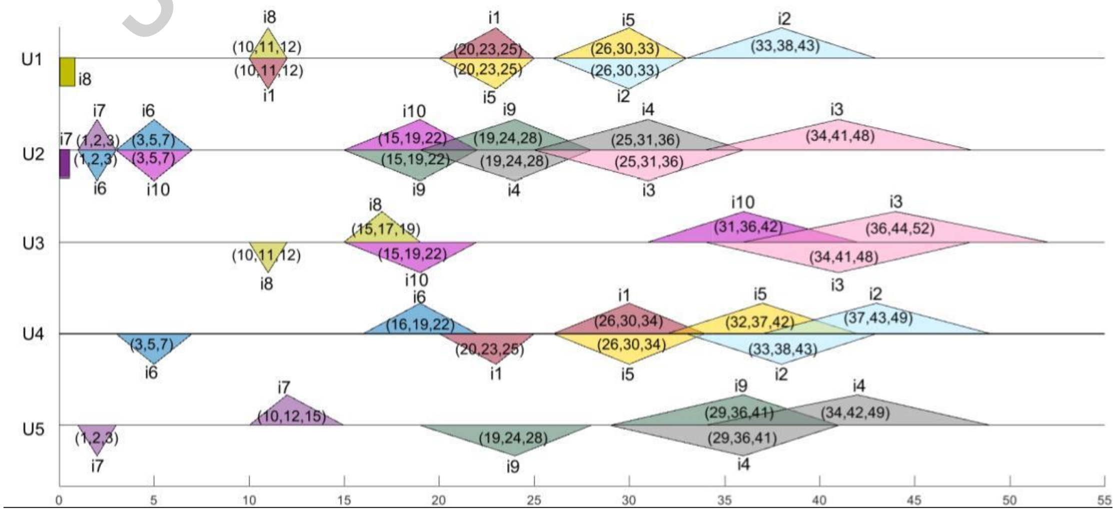
Journal Pre-proof

Journal Pre-proof

that DBSA-LS has adaptive, effective mutation and crossover operations for the FMMSP.

Table 7. Computational results of various algorithms on the medium and large-size problem.

Instance	Algorithm	Best value	Mean value	Worst value
o20s5u12	DBSA-LS	**(214, 236,238)**	**(224.00, 245.80,248.70)**	**(233, 257,260)**
	BDBSA	(230,256,259)	(236.05, 261.20,263.70)	(242, 266,273)
	MBSA	(244,267,272)	(247.00, 274.00,281.00)	(245.8,270.5,274.2)
	IGA	(224, 248,249)	(232.70, 256.10, 260.80)	(242, 267, 271)
o30s3u10	DBSA-LS	**(186,203,205)**	**(188.90, 208.10,216.00)**	(188.50, 219,229.5)
	BDBSA	(189, 210,212)	(193.80, 213.20,222.40)	**(199, 216,222)**
	MBSA	(203,220,221)	(202.60,223.70,231.50)	(208,228,229)
	IGA	(185, 204,207)	(191.10,209.50,221.90)	(200, 221,230)
o30s4u11	DBSA-LS	**(258,285,289)**	**(267.70,294.40, 299.60)**	**(278,306, 314)**
	BDBSA	(271,302, 306)	(279.30,308.30,312.30	(284,313,319)
	MBSA	(283, 312, 317)	(289.90,318.60,322.05)	(296,323,326)
	IGA	(266, 295, 301)	(279.30,307.00 310.60)	(288,313,316)
o30s5u12	DBSA-LS	**(260, 292, 294)**	**(269.8,300.5,311.2)**	**(289, 317, 322)**
	BDBSA	(277, 310, 313)	(278.8, 310.2, 317.7)	(280, 313, 318)
	MBSA	(272, 303, 307)	(284.7,317.3,319.9)	(291, 325,326)
	IGA	(266, 298, 309)	(278.55,308.40,312.95)	(292.5,317, 315.5)
o40s2u7	DBSA-LS	**(216.5,219,228.5)**	**(202.95, 221.90,250.65)**	**(206, 227, 262)**
	BDBSA	(197,218, 249)	(202.55,222.00,246.95)	(204, 226, 247)
	MBSA	(212, 231, 244)	(207.10, 228.90,257.40)	(209,229,262)
	IGA	(199,221, 248)	(200.40,221.70,253.80)	**(204, 225, 259)**
o40s3u8	DBSA-LS	**(268, 298, 308)**	**(275.90, 305.60,309.10)**	**(278,310,312)**
	BDBSA	(277, 307, 309)	(279.60, 309.60, 313.70)	(284,314, 315)
	MBSA	(291,321, 323)	(297.3,328, 331.7)	(299, 334,339)
	IGA	(275, 300, 303)	(279.90,308.20, 312.10)	(284,313, 316)
o40s4u12	DBSA-LS	**(301,337, 338)**	**(305.13,339,340.6)**	**(305.1,341.35,342.6)**
	BDBSA	(304,337,338)	(309,345,344)	(318,351,346)
	MBSA	(323,359,376)	(330.9,366.7, 370.3)	(333,371,372)
	IGA	(313,347,348)	(318.00,352.40,352.80)	(327,358,359)
o40s5u13	DBSA-LS	**(335,372,397)**	**(341.90,377.30, 417.40)**	**(360,396,397)**
	BDBSA	(336,375,433)	(355.18,393.1,409.1)	(355,392,438)
	MBSA	(360,389, 435)	(364,399.2, 429.7)	(362,405,442)
	IGA	(333,377,427)	(352.10, 389.00, 418.70)	(361,394,438)



Journal Pre-proof

Fig. 12. Fuzzy Gantt chart of instance o10s2u5.

The processing data of instance o10s2u5 is listed in Appendix Table A1. The fuzzy Gantt chart of o10s2u5 is shown in Fig. 12. In the figure, the TFN below the vertical coordinates represents the fuzzy starting time of orders for the unit and that above the vertical coordinates is the fuzzy completion time of orders for the unit. The TFN of the same order has the same color. The rectangle in the chart represents the fuzzy number (0,0,0).

Instances o10s3u8, o20s2u6 and o40s5u13 are selected to draw convergence curves for all of the algorithms in Figs. 13-15, based on the first ranking rule of the best objective values. These figures show that, when the same stopping criterion is employed, the proposed DBSA-LS algorithm outperforms the other three algorithms for the FMMSP to minimize the fuzzy makespan.

Journal Pre-proof

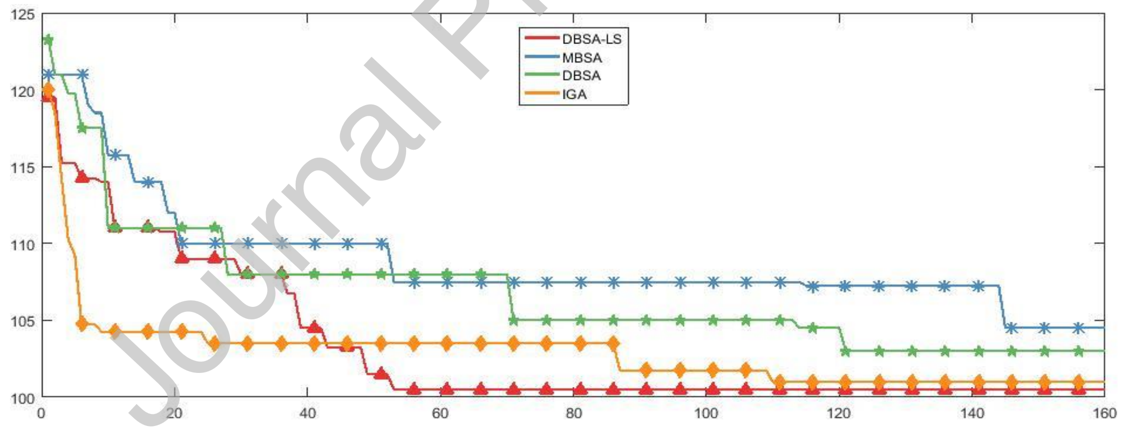


Fig. 13. Convergence curve of four algorithms for o10s3u8.

Journal Pre-proof

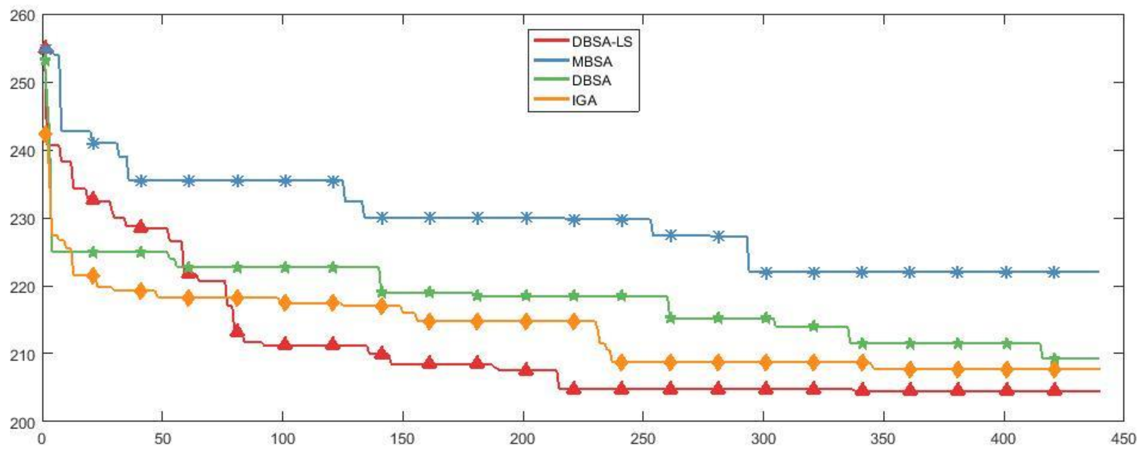


Fig. 14. Convergence curve of four algorithms for o20s2u6.

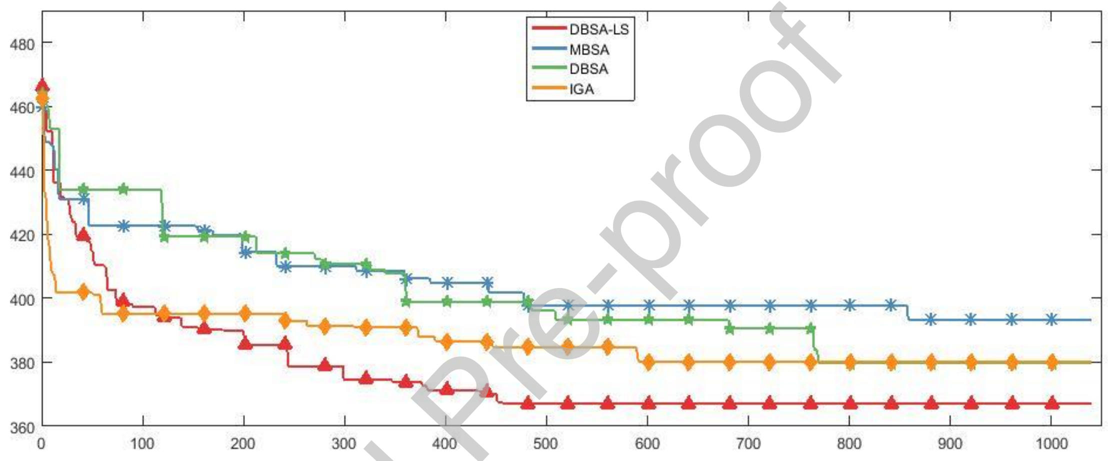


Fig. 15. Convergence curve of four algorithms for o40s5u13.

In summary, DBSA-LS presents excellent performance on sixteen different sizes of the FMMSP. This performance verifies its appropriate balance between exploitation and exploration due to the introduced improved strategies.

6. Conclusions and future work

This study proposes a discrete BSA with local search based on an insert operation to solve the FMMSP. A few new discrete operations are defined for the problem. When the improved algorithm generates the mutation individual, the information of the global best solution is introduced with the selection probability to improve the convergence speed of the proposed algorithm. To enhance the

Journal Pre-proof

Journal Pre-proof

exploitation ability of the improved algorithm, the insert local search algorithm is applied in the first stage of the Gbest solution of the population when the Gbest solution of the population has been non-improving in setting iteration times. Specific experimental tests are conducted to observe the influence of different parameter combinations on the performance of the proposed algorithm. The improved algorithm has an outstanding performance only if a good balance between the information diversity and the guide of the Gbest solution is realized. Comprehensive experimental tests are conducted to investigate the performance of DBSA-LS on the FMMSP, compared with three other comparison algorithms. The experimental results verify the superiority of DBSA-LS to the comparison algorithms. Thus, DBSA-LS is a promising candidate method to solve the FMMSP. In future studies, DBSA-LS could be combined with other works to solve the FMMSP with constraints, such as machine setup time and eligibility. The proposed algorithms can also be extended to the MMSP with different fuzzy parameters or optimization objectives.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No.61973120, 61573144, 61773165, 61673175), the Program of Introducing Talents of Discipline to Universities (the 111 Project) (Grant B17017) is also grateful appreciated. Fundamental Research Funds for the Central Universities (Grant No. 222201917006).

Declaration of interests

The authors declare that they have no known competing financial interests or personal

Journal Pre-proof

relationships that could have appeared to influence the work reported in this paper.

References

[1] F. Novara, G. Henning, A hybrid CP/MILP approach for big size scheduling problems of multiproduct, multistage batch plants, Comput. Aided Chem. Eng. 37 (2015) 2027–2032. https://doi.org/10.1016/B978-0-444-63576-1.50032-7.

[2] S. Moniz, A.P. Barbosa-Póvoa, et al, New general discrete-time scheduling model for multipurpose batch plants, Ind. Eng. Chem. Res. 52 (48) (2013) 17206-17220. https://doi.org/10.1021/ie4021073.

[3] A.F. Merchan, H. Lee, C.T. Maravelias, Discrete-time mixed-integer programming models and solution methods for production scheduling in multistage facilities, Comput. Chem. Eng. 94 (2016) 387-410. https://doi.org/10.1016/j.compchemeng. 2016.04.034.

[4] H. Lee, C.T. Maravelias, Mixed-integer programming models for simultaneous batching and scheduling in multipurpose batch plants, Comput. Chem. Eng. 106 (2017) 621-644. https://doi.org/10.1016/j.compchemeng.2017.07.007.

[5] H. Lee, C.T. Maravelias, Discrete-time mixed-integer programming models for short-term scheduling in multipurpose environments, Comput. Chem. Eng.107 (2017) 171-183. https://doi.org/10.1016/j.compchemeng.2017.06.013.

Journal Pre-proof

Journal Pre-proof

[6] A. Sundaramoorthy, C.T. Maravelias, Simultaneous batching and scheduling in multistage multiproduct processes, Ind. Eng. Chem. Res. 47 (5) (2008) 1546-1555. https://doi.org/10.1021/ie070944y.

[7] P.A. Marchetti, J. Cerdá, Efficient precedence-based multistage batch scheduling formulation with nontrivial tightening constraints, Comput. Aided Chem. Eng. 40 (2017) 1429-1434. https://doi.org/10.1016/B978-0-444-63965-3.50240-3.

[8] P.A. Marchetti, J. Cerdá, A general resource-constrained scheduling framework for multistage batch facilities with sequence-dependent changeovers, Comput. Chem. Eng. 33 (4) (2009) 871-886. https://doi.org/10.1016/j.compchemeng.2008.12.007.

[9] J.M. Pinto, I.E. Grossmann, A continuous time mixed integer linear programming model for short term scheduling of multistage batch plants, Ind. Eng. Chem. Res. 34 (9) (1995) 3037-3051. https://doi.org/10.1021/ie00048a015.

[10] N. Lamba, I.A. Karimi, Scheduling parallel production lines with resource constraints. 1. model formulation, Ind. Eng. Chem. Res. 41 (4) (2002) 779-789. https://doi.org/10.1021/ie010009p.

[11] A. Sundaramoorthy, I.A. Karimi, A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants, Chem. Eng. Sci. 60 (10) (2005) 2679-2702. https://doi.org/10.1016/j.ces.2004.12.023.

[12] I. Harjunkoski, I.E. Grossmann, Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods,

Journal Pre-proof

Comput. Chem. Eng. 26 (11) (2002) 1533-1552. https://doi.org/10.1016/S0098-1354 (02)00100-X.

[13] G. Deng, H. Yang, S. Zhang, An enhanced discrete artificial bee colony algorithm to minimize the total flow time in permutation flow shop scheduling with limited buffers, Math. Probl. Eng. 2016 (2016) 1-11. https://doi.org/10.1155/2016/ 7373617.

[14] T.L. Lin, S.J. Horng, et al, An efficient job-shop scheduling algorithm based on particle swarm optimization, Expert Syst. Appl. 37 (3) (2010) 2629-2636. https:// doi.org/10.1016/j.eswa.2009.08.015.

[15] K. Fan, M. Wang, et al, Scatter search algorithm for the multiprocessor task job-shop scheduling problem, Comput. Ind. Eng. 127 (2019) 677-686. https:// doi.org/10.1016/j.cie.2018.11.006.

[16] Y. He, C.W. Hui, Genetic algorithm for large-size multi-stage batch plant scheduling, Chem. Eng. Sci. 62 (5) (2007) 1504-1523. https://doi.org/10.1016/ j.ces. 2006.11.049.

[17] Y. He, C.W. Hui, A novel search framework for multi-stage process scheduling with tight due dates, AIChE J. 56 (8) (2010) 2103 – 2121. https://doi.org/10.1002/ aic. 12134.

[18] B. Shi, X. Qian, et al, Rule-based scheduling of multi-stage multi-product batch plants with parallel units, Chinese J. Chem. Eng. 25 (8) (2017) 1022-1036. https://

Journal Pre-proof

Journal Pre-proof

doi.org/10.1016/j.cjche.2017.03.014.

[19] R. E. Bellman and L. A. Zadeh, Decision-making in a fuzzy environment, Manage. Sci. 17 (4) (1970) 141-164. https://doi.org/10.1287/mnsc.17.4.b141.

[20] P. Civicioglu, Backtracking search optimization algorithm for numerical optimization problems, Appl. Math. Comput. 219 (15) (2013) 8121-8144. https:// doi.org/10.1016/j.amc.2013.02.017.

[21] F. Zou, D. Chen, et al, Community detection in complex networks: multi-objective discrete backtracking search optimization algorithm with decomposition, Appl. Soft Comput. J. 53 (2017) 285–295. https://doi.org/10.1016/ j.asoc.2017.01.005.

[22] S. Vitayasak, P. Pongcharoen, C. Hicks, A tool for solving stochastic dynamic facility layout problems with stochastic demand using either a genetic algorithm or modified backtracking search algorithm, Int. J. Prod. Econ. 190 (2017) 146-157. https://doi.org/10.1016/j.ijpe.2016.03.019.

[23] Q. Lin, L. Gao, X. Li, C. Zhang, A hybrid backtracking search algorithm for permutation flow-shop scheduling problem, Comput. Ind. Eng. 85 (2015) 437-446. https://doi.org/10.1016/j.cie.2015.04.009.

[24] J. Lin, Z.J. Wang, X. Li, A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem, Swarm Evol. Comput. 36 (2017) 124-135. https://doi.org/10.1016/j.swevo.2017.04.007.

Journal Pre-proof

Journal Pre-proof

[25] J. Lin, Backtracking search based hyper-heuristic for the flexible job-shop scheduling problem with fuzzy processing time, Eng. Appl. Artif. Intell. 77 (2019) 186–196. https://doi.org/10.1016/j.engappai.2018.10.008.

[26] K. Yu, J.J. Liang, et al, Multiple learning backtracking search algorithm for estimating parameters of photovoltaic models, Appl. Energy. 226 (2018) 408-422. https://doi.org/10.1016/j.apenergy.2018.06.010.

[27] Y. Liu, I.A. Karimi, Scheduling multistage, multiproduct batch plants with nonidentical parallel units and unlimited intermediate storage, Chem. Eng. Sci. 62 (2007) 1549–1566. https://doi.org/10.1016/j.compchemeng.2007.02.002.

[28] Y. Xue, J. Yuan, Scheduling model for a multi-product batch plant using a pre-ordering approach, Chem. Eng. Technol. 31 (3) (2008) 433–439. https://doi: 10.1002/ceat.200700168.

[29] E. Taillard, Some efficient heuristic methods for the flow shop sequencing problem, Eur. J. Oper. Res. 47 (1990) 65-74. https://doi.org/10.1016/ 0377-2217(90) 90090-X.

[30] T. Schiavinotto, T. Stützle, A review of metrics on permutations for search landscape analysis, Comput. Oper. Res. 34 (10) (2007) 3143-3153. https://doi.org/ 10.1016/j.cor.2005.11.022.

[31] D.C. Montgomery, Design and analysis of experiments, 8th ed., JohnWiley and Sons, Inc, NewYork, 2013. https://doi.org/10.1002/9781118147634.

Journal Pre-proof

Journal Pre-proof

[32] C. Yu, Q. Semeraro, A. Matta, A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility, Comput. Oper. Res. 100 (2018) 211–229. https://doi:10.1016/j.cor.2018.07.025.

Appendix A

Table A1. Processing data of instance o10s2u5.

Order	Unit 1	Unit 2	Unit 3	Unit 4	Unit 5
	Stage 1		Stage 2		
1	(10,12,13)	(11,12,14)	(9,10,12)	(6,7,9)	(8,9,10)
2	(7,8,10)	(8,9,10)	(5,6,8)	(4,5,6)	(4,5,6)
3	(10,11,12)	(9,10,12)	(2,3,4)	(5,6,8)	(5,6,7)
4	(8,9,10)	(6,7,8)	(7,8,9)	(4,5,6)	(5,6,8)
5	(6,7,8)	(8,9,10)	(4,5,6)	(6,7,8)	(6,7,9)
6	(4,5,6)	(2,3,4)	(15,16,19)	(13,14,15)	(15,16,20)
7	(11,13,15)	(1,2,3)	(11,13,14)	(10,11,13)	(9,10,12)
8	(10,11,12)	(18,19,23)	(5,6,7)	(6,7,9)	(7,8,10)
9	(5,6,8)	(4,5,6)	(14,15,16)	(19,21,25)	(10,12,13)
10	(15,17,20)	(12,14,15)	(16,17,20)	(17,18,21)	(18,19,21)

Journal Pre-proof

Xueli Yan was born in 1986. She received her master's degree in automation from East China University of Science and Technology in 2012. She is pursuing the Ph.D. degree in Department of Automation, ECUST. Her main search area includes intelligent optimization and scheduling methods of batch plants in uncertainty circumstance.

Journal Pre-proof

Yuxin Han was born in 1990. He received the B.E. degree in automation from East China University of Science and Technology in 2013. He is pursuing the Ph.D. degree in Department of Automation, ECUST. His main search area includes intelligent optimization and scheduling methods of batch plants.

Xingsheng Gu received the B.S. degree from Nanjing Institute of Chemical Technology in 1982, M.S. and Ph. D. degree from East China University of Chemical Technology in 1988 and 1993, respectively. He is currently a professor at School of Information Science and Engineering, East China University of Science and Technology. His research interests include planning and scheduling for process industry, modeling, control and optimization for industry processes, intelligent optimization, faults detection and diagnosis, etc.

Journal Pre-proof