Volume: 11 Issue: VI Month of publication: June 2023
DOI: https://doi.org/10.22214/ijraset.2023.54482

Hybrid Enhanced Evolutionary Algorithm with Differential Evolution for Distributed No-Wait Flow-Shop Scheduling Problem

Kai Steven Ellie[1], Wenqiang Zhang[2]
[1, 2]College of Information Science and Technology, Henan University of Technology, Zhengzhou, 450001, China

Abstract: The distributed no-wait flow-shop scheduling problem (DNWFSP) holds significant importance in real-world production environments. However, existing algorithms for solving the DNWFSP have limitations such as unsatisfactory solutions. Therefore, this paper proposes a hybrid enhanced multi-objective evolutionary algorithm (HEMOEA) with differential evolution (HEMOEA-DE) to solve DNWFSP with the criteria of minimizing makespan and total processing time. The algorithm incorporates a mathematical model for the DNWFSP. Effective encoding and decoding methods are employed, and DE is utilized as an efficient genetic operator for chromosome recombination. The HEMOEA-DE algorithm combines the Pareto dominating, dominated relationship-based fitness function and vector-evaluated genetic algorithm algorithms within HEMOEA to approach the Pareto Front quickly. To evaluate the performance of the proposed method, experiments were conducted on the E. Taillard benchmark problem. The outcomes were compared with those of other well-known algorithms. Regarding convergence and performance on the DNWFSP, the HEMOEA-DE algorithm outperformed other HEMOEA algorithms.
Keywords: hybrid enhanced evolutionary algorithm; distributed no-wait flow-show scheduling problem, differential evolution

I. INTRODUCTION

Production scheduling is crucial in manufacturing systems, enhancing production efficiency and resource utilization [1]. It involves assigning limited production resources to tasks within a specific timeframe to optimize the objectives by decision-makers [2]. With manufacturing technology and equipment advancements, production scheduling has become more complex. Additionally, due to globalization, single factories can no longer adapt effectively to market demand changes [3]. Consequently, the structure of factories has shifted from centralized to decentralized, leading to the popularity of the multi-shop co-production model. Distributed manufacturing offers benefits such as resource sharing, cost reduction, risk mitigation, improved product quality, enhanced economic outcomes, and increased responsiveness to market changes [4]. Among the various problems associated with distributed manufacturing, the distributed no-wait flow-shop scheduling problem (DNWFSP) has received significant attention and has practical applications [5].

The DNWFSP with the makespan criterion is currently the most extensively studied due to its inherent complexity [6]. The DNWFSP is a more general form of the flow-shop scheduling problem (FSP) and comprises two interrelated sub-problems [7]. The first sub-problem involves assigning jobs to factories, while the second sub-problem entails arranging the job sequences within each factory. These two sub-problems are interconnected, and their optimization requires simultaneous consideration [8]. However, optimizing each sub-problem independently does not guarantee the optimal solution for the coupled problem. Therefore, the existing research on single-shop scheduling problems cannot be directly applied to the DNWFSP. Moreover, from a mathematical perspective, the DNWFSP is classified as an NP-hard problem, and incorporating sub-problems expands the solution space, further intensifying the problem's difficulty. In conclusion, the DNWFSP presents significantly more challenges than single-shop scheduling, and its exploration holds substantial research significance, necessitating the development of novel methodologies.

Currently, a growing range of approaches is available for solving the DNWFSP. Traditional mathematical techniques like enumeration and branch and bound are suitable for smaller and simpler problems but become computationally expensive for more significant instances. As a result, heuristics and meta-heuristics have gained prominence in solving complex combinatorial optimization problems. Over the years, various optimization algorithms have been developed and applied to the DNWFSP, including genetic algorithms (GA) [9], tabu search [12], distribution estimation algorithm [13], hybrid immune algorithm [14], scatter search [15], chemical reaction optimization [16], iterative greedy algorithm [17], artificial bee colony algorithm (ABC) [18], water wave optimization algorithm [19], and cooperative optimization algorithm [20].

These algorithms operate differently from traditional mathematical methods, leveraging unique theories and mechanisms to balance exploration and exploitation during the iterative process. Furthermore, some studies have also explored addressing additional objectives of the DNWFSP, such as the total processing time criterion [21].

Two different industrial indicators have been suggested as a solution to the interconnected sub-problems presented by the DNWFSP [21]. The first indicator, makespan, has been extensively studied [22]. The second indicator focuses on minimizing the total processing time. Reducing the total processing time can decrease the number of jobs in progress and promote stable resource utilization. This advantage becomes particularly significant in distributed environments where resource imbalances and varying job progress can arise due to multiple factories.

The entire processing time has been acknowledged as a very relevant and crucial aim in today's evolving industrial environment. Therefore, this study aims to tackle the DNWFSP by minimizing the makespan and total processing time criteria. By explicitly modeling the problem to emphasize one of these two indicators, it becomes more convenient to solve the DNWFSP. However, this approach can only focus on optimizing one indicator, making it challenging to optimize the other. Another method that can be employed to address these closely related problems is to randomly arrange the process sequence of each job and assign them randomly to factories [16]. This approach allows for considering both the makespan and total processing time indicators. Hence, this method is employed in solving the DNWFSP.

This research presents a novel approach called hybrid enhanced multi-objective evolutionary algorithm (HEMOEA) with differential evolution (HEMOEA-DE) for addressing optimization problems. HEMOEA-DE combines HEMOEA and DE strategies to enhance the algorithm's convergence performance, distribution performance, and diversity. The enhanced direction search strategy focuses on exploring multiple directions of the Pareto front (PF) within, aiming to expedite convergence in the upper, central, and lower regions of the PF. This strategy improves the algorithm's capability to discover optimal solutions across the entire PF. Combining these two strategies further improves convergence performance, distribution performance, and diversity. A two-vector representation method is devised for the individuals within the enhanced direction search strategy to facilitate the encoding and decoding processes.

This method enables efficient updates of using exchange sequences, crossover, and mutation operations. The performance of the proposed HEMOEA-DE algorithm is compared against several other widely used multi-objective optimization algorithms, including NSGA-II, SPEA2, HEMOEA, MOEA/D. Experimental results demonstrate that HEMOEA-DE exhibits excellent convergence performance while ensuring high-quality solutions with good distribution performance and diversity.

The rest of this research is structured as follows. In Section 2, we present relevant related work regarding the resolution of the DNWFSP. Section 3 introduces the formulation of the DNWFSP. The HEMOEA-DE algorithm is described in Section 4. Section 5 presents and discusses the experimental data of HEMOEA-DE. Lastly, Section 6 concludes this study and discusses future work.

II. RELATED WORK

Recent research has focused on the DNWFSP and the development of hybrid enhanced evolutionary algorithms to address this problem. The DNWFSP aims to optimize multiple criteria while ensuring uninterrupted processing of each job across all machines. However, it becomes highly NP-hard when more than two machines are involved [23]. To improve the effectiveness of the DNWFSP, several hybrid evolutionary algorithms have been proposed.

Ishibuchi et al. suggested combining local search with evolutionary multi-objective optimization (EMO) techniques to enhance the performance of the DNWFSP [25]. Zhao et al. recommended utilizing a hybrid biogeography with dynamic surroundings search to improve the neighbourhood makespan criterion of DNWFSP [26]. Orhan et al. developed a hybrid ant colony technique based on the DNWFSP crossover and mutation process to reduce the maximum completion time [27]. Qian et al. introduced the discrete differential evolution (DDE) method, which utilizes distinct work shifts and improved mutation and crossover operators to represent people [28]. Zheng et al. presented the hybrid quantum-inspired evolutionary algorithm (HQEA) to determine the order of tasks by encoding the quantum chromosome using a quantum rotation angle [29]. Imran et al. proposed a Genetic Algorithm (GA) applied in a tabular context to reduce the overall flowtime of the DNWFSP [30]. Qu et al. suggested a flower pollination algorithm based on hormone regulation mechanisms to reduce the longest possible completion time of the DNWFSP [31]. Shao et al. proposed a hybrid discrete optimization approach incorporating a teaching-probabilistic learning mechanism to reduce the makespan for the DNWFSP [32].

In addition to the DNWFSP, researchers have also addressed the Permutation Flow-Shop problem (PFSP), which involves preparation and transportation times simultaneously. Jiang and Wang improved the MOEA/D algorithm for solving two-objective PFSPs [33].

Hybrid enhanced evolutionary algorithms (HEEA) have been developed, integrating tabu search into evolutionary algorithms and possessing unique features such as the longest recombination operator based on universal sequences and a population update standard based on similarity and quality [34].

M.R. Amin et al. proposed three methods for solving the DNWFSP: genetic algorithm, bottleneck-based heuristic algorithms, and simulated annealing [24]. Shao et al. created a hybrid meta-heuristic based on a teaching-learning mechanism to solve the DNWFSP [35]. Bewoor et al. proposed an efficient combined Particle Swarm Optimization (PSO) algorithm for resolving the DNWFSP [36]. Chiang et al. solved the multi-objective PFSP using an upgraded version of MOEA/D [37,38]. Wang et al. combined local search methods with NSGA-II and a memetic algorithm based on MOEA to solve the parallel FSP [39]. Nouha et al. suggested a GA based on NSGA-II locate the Pareto-optimal frontier for the permutation FSP [40]. Wei et al. developed a bi-criteria permutation FSP algorithm using NSGA-II-MOLS [41]. Zhang et al. presented a HEMOEA to reduce makespan and total flow time (TFT) for bi-criteria FSP [42]. The MOHEEA method employs the VEGA and PDDR-FF sampling techniques to investigate the Pareto front [43]. A novel sample method based on PDDR-FF is used by the proposed bi-criteria FSP algorithm based on HEMOEA to enhance performance [44]. The algorithm's exclusive repair approach based on PDDR-FF and its choice of approach in accordance with VEGA [45,46].

It is important to note that the algorithms mentioned above have been developed for solving different types of DNWFSPs, each with specific objective functions. However, to provide a comprehensive review, it is necessary to consider all the multi-objective optimization algorithms developed for solving DNWFSPs. As for the best-performing algorithm for solving DNWFSPs, without specific information on the latest developments in the field, it is challenging to pinpoint a single algorithm that outperforms all others. The effectiveness of an algorithm can vary depending on the problem instance and the specific objectives considered. However, it is worth mentioning that multi-objective evolutionary algorithms (MOEAs) have been widely used and proven effective in solving DNWFSPs.

MOEAs, such as Non-dominated Sorting Genetic Algorithm II (NSGA-II) [40] and Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D) [33], have been successfully applied to tackle various DNWFSP variants. These algorithms use evolutionary principles, including population-based search, selection, and recombination, to explore the Pareto-optimal solutions representing trade-offs between conflicting objectives. Without specific information about the latest advancements, NSGA-II and MOEA/D can be considered robust baseline algorithms for solving DFSPs. These algorithms have demonstrated exemplary performance and have been extensively used in the literature to obtain Pareto-optimal solutions for different objective functions in DNWFSPs.

The field of DNWFSP research is continuously evolving, and new algorithms or hybrid approaches may emerge as more effective alternatives. To stay updated with the current state-of-the-art and identify the best-performing algorithm for a specific DNWFSP variant and objective function, it is recommended to refer to recent publications and research advancements in the field.

III. PROBLEM FORMULATION

In the DNWFSP, a set of n jobs must be completed on m machines. Every machine j has a processing period for each task i. A further restriction of the DNWFSP is "no-wait," which requires that once a task begins processing on one computer, it must continue uninterruptedly on all m workstations. The tasks are first postponed in order to adhere to this constraint. Consequently, a delay matrix has to be calculated in order to address this issue [47].

The research assumes a number of assumptions, including the absence of task pre-emption during processing and the absence of any further restrictions on the machines' capacity or processing sequence. The research also presupposes that the goal is to reduce the overall work completion time, or the time required for the last task to be finished on each machine. Additionally, the research sees the problem as a multi-objective optimization challenge that seeks to cut down on the overall time required to complete the jobs and their ideal level of tardiness.

These are some other hypotheses in this research:

1) The calculation's duration is predetermined;
2) Once a job has started, it can never be halted;
3) The machine can only do one task at a time;
4) Each task is currently handled by a single machine;
5) There is no machine preparation time to prepare to perform activities;
6) The period required for moving jobs among machines is insignificant.

The following is a definition of the DNWFSP's mathematical formulation:

m = The number of different machines

n = The number of jobs;

$P_{(i,j)}$- Time required for executing; j- machine; i- job;

δ *(i, s)* - the shortest possible wait on the initial machine during the commencement *of jobs i* and *s;*

$\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$- a a series of n tasks to be completed on m machines;

δ (σ_{i-1}, σ_i) – the minimum delay between the sequence $(i-1)$ on the first machine and the start of two sequential operations from position i*;*

p (σ_i, j) - processing time (a job at position i in the given sequence on machine j);

C (σ_i) - completion time of the job in the i_{th} position of a given sequence;

δ (i, k) - delay matrix;

For j = 1, 2, . . ., m and i = 1, 2…, n.

$$C(\sigma_1) = \sum_{j=1}^{m} p(\sigma_1, j) \qquad (3\text{-}1)$$

$$C(\sigma_2) = \delta(\sigma_1, \sigma_2) + \sum_{j=1}^{m} p(\sigma_2, j) \qquad (3\text{-}2)$$

$$C(\sigma_i) = \sum_{k=2}^{i} \delta(\sigma_{k-1}, \sigma_k) + \sum_{j=1}^{m} p(\sigma_i, j) \qquad (3\text{-}3)$$

Using the expression, calculate the total processing time of every job i (i = 1, 2, 3... n) on the given m machines.

The following is the formula for optimizing total time needed for processing:

$$\text{min. TPT} = P_i \qquad (3\text{-}4)$$

$$P_i = \sum_{j=1}^{m} p\,(\sigma_i, j) \qquad (3\text{-}5)$$

The following are the mathematical equations for minimizing maximum flow time (makespan):

$$\text{min. makespan} = \sum_{i=1}^{n} C\,(\sigma_1) \qquad (3\text{-}6)$$

$$= \sum_{i=2}^{n} \{\sum_{k=2}^{i} \delta(\sigma_{k-1}, \sigma_k) + \sum_{j=1}^{m} p(\sigma_i, j)\} + \sum_{j=1}^{m} p(\sigma_1, j) \qquad (3\text{-}7)$$

$$= \sum_{i=2}^{n} (n+1-i)\delta(\sigma_{i-1}, \sigma_i) + \sum_{i=2}^{n} \sum_{j=1}^{m} p(i, j) \qquad (3\text{-}8)$$

$$\delta(i,k) = p(i,1) + \max\{\sum_{h=2}^{r} p(i,h) - \sum_{h=1}^{m} p(k,h), 0\}; 2 \le r \le m \qquad (3\text{-}9)$$

A delay matrix of dimensions $n \times n$ contains all the values δ (i, k) between two consecutive jobs i and k (where $i \neq k$). These values determine the objective function's value in a sequence of n jobs. Using a matrix of dimensions $n \times m$ with processing times $p(\sigma_i, j)$, we can generate a set of n! solutions, denoted by F(σ). From this set, we can identify the optimal sequence, $F(\sigma^*)$, which can be determined using the equation provided.

To explain, the delay matrices provide details regarding the interval between two consecutive tasks, and the processing time matrices provides details about how long each job takes to finish.

By combining these two matrices, we can generate a set of solutions, each representing a different ordering of jobs. There is n! possible solutions, which can be represented by F(σ).

We can identify the optimal sequence from this set of solutions, denoted by $F(\sigma^*)$. The equation provided can be used to determine $F(\sigma^*)$ based on the values in the processing time matrix and the delay matrix. By identifying the optimal sequence, we can minimize the job sequence's overall efficacy and its ultimate function:

$$F(\sigma^*) \leq F(\sigma) \tag{3-10}$$

This research utilizes two objectives to optimize the job sequence: minimizing the total processing time (3-11) and minimizing the makespan (3-12).

$$min.\, f_1(X) = \sum_{j=1}^{m} p\,(\sigma_i, j) \tag{3-11}$$

$$min.\, f_2(X) = \sum_{i=1}^{n} C\,(\sigma_1) \tag{3-12}$$

The first objective is to reduce the time needed to do each job in succession. The second objective, on the other hand, focuses on reducing the amount of exertion required to perform the longest task in the series (i.e., the makespan). Both of these objectives are important in optimizing job scheduling, as they can impact the overall efficiency and productivity of the process.

The research uses these two objectives to identify the optimal job sequence that reduces the processing time and the makespan. By analyzing the information, this may be accomplished. Provided in the processing time and delay matrix and generated solutions representing different job sequences. We can ensure that the job sequence is efficient and effective by selecting a solution that satisfies both objectives.

IV. THE PROPOSED METHOD

A. Overview of the Hemoea-DE

HEMOEA-DE operates in two sub-groups: PDDR-FF fitness function and VEGA edge selection strategy. The VEGA edge selection strategy selects superior chromosomes for two single objectives, while PDDR-FF chooses the Pareto front's center region. These sub-groups excel in their areas of objective one, two, and Pareto Frontier. The population size is used to determine the size of the sub-groups, eliminating the need for chromosome screening. The subgroups are updated in different directions based on their performance.

Those performing well on objective are updated towards objective one, on objective two in their respective area, and those near the Pareto center in that region. DE strategy is applied to a temporary chromosome created by mixing updated chromosomes of the same size as the defined chromosome.

DE generates multiple differential chromosomes from two randomly selected chromosomes, and excellent chromosomes are chosen from the differential and temporary chromosomes as new chromosomes in the next generation. DE quickens the algorithm's approach toward the Pareto Front. HEMOEA-DE effectively integrates the basic framework of DE and HEMOEA, leveraging the advantages and eliminating the disadvantages.

Algorithm-1 provides the computational process of HEMOEA-DE's pseudo-code.

To solve the DNWFSP in a multi-factory situation, a hybrid enhanced evolutionary algorithm called HEMOEA-DE has been developed. By integrating the basic framework of DE and HEMOEA, HEMOEA-DE may hasten the algorithm's approach to the Pareto Front take full advantage of both frameworks while overcoming their limitations. Moreover, it has seen notable advancements in the formula of the HEMOEA, which is designed for discrete permutation and combination problems. The HEMOEA-DE algorithm as seen in Figure 4.1, consists of four major phases, and its pseudo-code is presented in Algorithm 1.

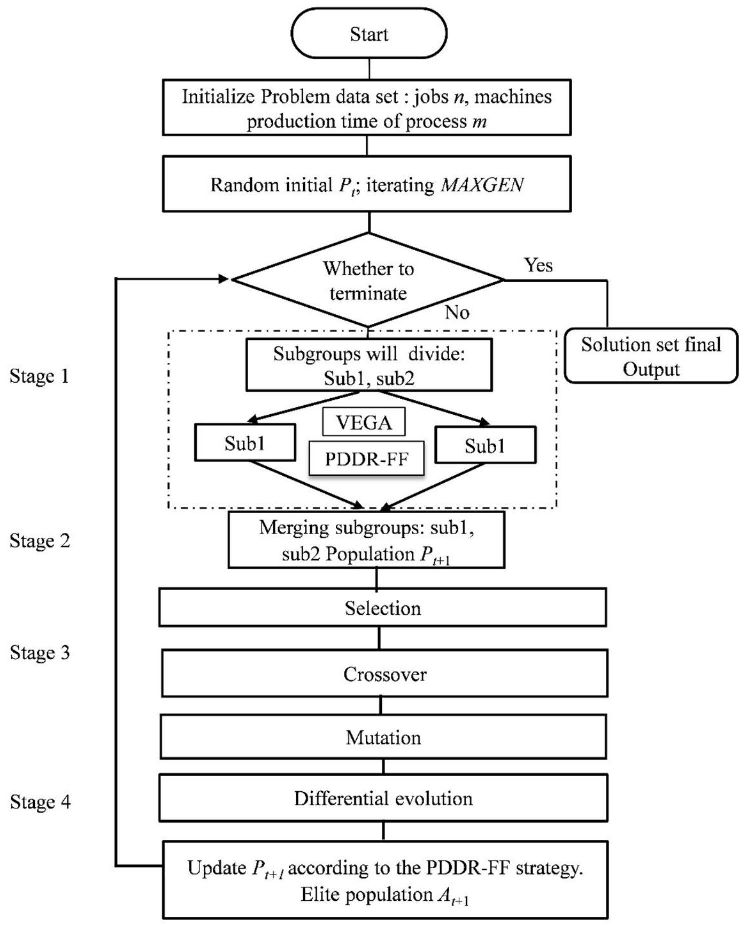
Start
Initialize Problem data set : jobs n, machines production time of process m
Random initial P_t; iterating $MAXGEN$
Whether to terminate
Yes
No
Solution set final Output
Stage 1
Subgroups will divide: Sub1, sub2
VEGA
PDDR-FF
Sub1
Sub1
Stage 2
Merging subgroups: sub1, sub2 Population P_{t+1}
Selection
Stage 3
Crossover
Mutation
Stage 4
Differential evolution
Update P_{t+1} according to the PDDR-FF strategy. Elite population A_{t+1}

Fig. 4.1 Basic framework of HEMOEA-DE

HEMOEA-DE operates in two sub-groups: PDDR-FF fitness function and VEGA edge selection strategy. The VEGA edge selection strategy selects superior chromosomes for two single objectives, while PDDR-FF chooses the central area of the Pareto front. These sub-groups excel in their areas of objective one, two, and Pareto Frontier. The sub-group size is set to the population size, eliminating the need for chromosome screening. The subgroups are updated in different directions based on their performance. Those performing well on objective one are updated towards objective one, those on objective two in their respective area, and those near the Pareto centre in that region. DE strategy is applied to a temporary chromosome created by mixing updated chromosomes of the same size as the defined chromosome. DE generates multiple differential chromosomes from two randomly selected chromosomes, and excellent chromosomes are chosen from the differential and temporary chromosomes as new chromosomes in the next generation. DE quickens the algorithm's approach toward the Pareto Front. HEMOEA-DE effectively integrates the basic framework of DE and HEMOEA, leveraging the advantages and eliminating the disadvantages. Algorithm-1 provides the of computational process HEMOEA-DE's pseudo-code.

Algorithm 1. Repairing Procedure Pseudo-code (HEMOEA-DE for DNWFSP)

```
Input: a DNWFSP data set (one for each instance). Variable of the algorithm: PopSize,
    MaxGenerationsSize, Elite populationSize, quantity of machines $m$, number of tasks $n$, and process
    production times
Output: Set of Pareto solutions;
1. $t = 0$;
2. Reset the population's $P_t$;
3. Determine each individual's functioning objective result;
4. Determine the individual's PDDR-FF result;
```

```
5. To begin the elite population size, use the center of the hybrid sampling approach for the sample $P_t$.
$A_t$;
6 while ($t$ <=MaxGenerationsSize) do
7.     By coding ordinary modes, compute the objective $f_i(x)$ of $x$, $i$=1,2.
8.     eval($x$) fitness coefficient computation using normal mode decoding;
9.     divide the $P(t)$ into two subs using VEGA and PDDR-FF;
10.    $P(t+1)$ is formed by combining genes from two subgroups;
11.    For population $P_t$, use edge area DE to construct subpopulation 2 SP2;
12.    Create a mating pool by combining SP1, SP2, and $A_t$;
13.    Determine the significance of the individual goal variable;
14.    Combine and to make a temporary;
15.    Use a central area sampling approach for updating the elite group;
16.      $t = t + 1$;
17.                end while
18. Choose non-dominant individuals from $A_t$ to create the Pareto response set.
```

B. Encoding and decoding

1) Encoding method

Sometimes, the solution to a real-world problem cannot be applied directly to an algorithm. It may need to be preprocessed or transformed to a format the algorithm can accept. Additionally, the choice of encoding method can significantly impact the development of the algorithm. A suitable encoding method can simplify the decoding process, while a complex encoding method can increase the decoding time. The process of transforming the problem into a format acceptable to the algorithm is called encoding.

Moreover, it may also be defined as the problem's illustration. For the DNWFSP, a fixed-length sequence coding method has been employed based on the problem's unique characteristics. In this encoding method, each chromosome comprises a fixed-length sequence of real numbers, and each bit in the chromosome represents the job's serial number.

2) Decoding method

Decoding is the process of transforming the outcome space back into the problem space. The fixed-length encoding method can reflect the problem and meet the algorithm requirements, and the decoding process will be much easier. When using HEMOEA-DE to solve the DNWFSP, after determining the chromosome sequence, the formula can be used to calculate the processing time of DNWFSP so that two objective values can be obtained to assess the quality of each chromosomal generation. Simultaneously, decoding is rather straightforward since the most important aspect of the DNWFSP is to organize the serial numbers of processing period. The initial phase of the technique comprises encoding and decoding in order to tackle compatibility problems [48]. A two-vector technique is specifically used to express a solution for the DFSP. The job sequence (JS) vector indicates the order of tasks to be performed. The additional vector, factory assignment (FA), reflects each job's assignment to a single factory. The encoding approach used to solve the DNWFSP includes building an individual out of two vectors. The first vector is a list of six occupations, with each element representing a different employment. At the same time, the remaining vector, which has just two components, indicates each job's assignment to one of the two factories. Through this encoding method, two major problems of the DFSP can be effectively addressed. Moreover, manipulating these two vectors enables the individual to achieve both the job assignment and the sorting of jobs within each factory [49]

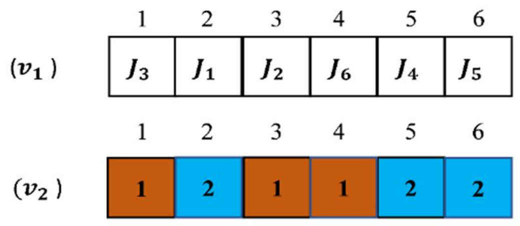

Fig. 4.2 genetic representation

The decoding process is simple when the encoding method described above is used. The total job sequence makes up the first vector. The jobs should be sorted according to the first vector after being first selected from each factory's jobs based on the second vector during decoding. The machine's processing schedule and the entire processing flow can be estimated once the sequence has been determined. Figure 4.3 shows two job sequences that will be obtained after encoding, each representing a factory's JS, so the problem's two objective values can finally be calculated.

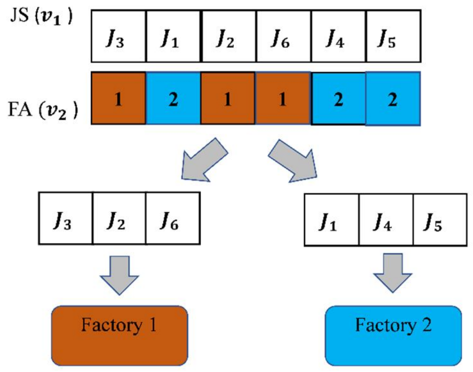

Fig. 4.3 Encoding scheme

C. *Global Search Strategy*

Searching for optimal solutions in multi-objective optimization problems often involves exploring the Pareto front, representing a set of trade-off solutions. This abstract introduces a global search strategy called hybrid sampling that combines two sampling techniques, namely Pareto dominating, dominated relationship-based fitness function (PDDR-FF) and vector-evaluated genetic algorithm (VEGA), to search the Pareto front efficiently.

1) *PDDR-FF and VEGA*

The proposed hybrid sampling strategy offers an efficient and balanced approach to globally searching for Pareto solutions. By integrating the PDDR-FF and VEGA techniques, the strategy effectively explores the core and edge regions, providing robust solutions. The selecting mechanism's computation technique is illustrated in Figure 4.4, providing a visual representation of the strategy's operation. The hybrid sampling strategy demonstrates its potential in solving complex multi-objective optimization problems by efficiently searching for diverse and high-performing solutions on the Pareto front.

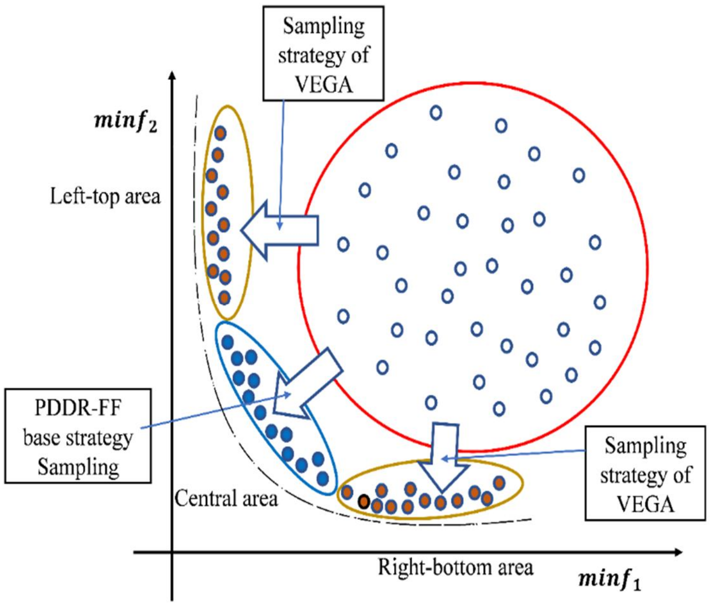
Sampling strategy of VEGA
$minf_2$
Left-top area
PDDR-FF base strategy Sampling
Central area
Sampling strategy of VEGA
Right-bottom area
$minf_1$

Fig. 4.4 Hybrid sampling strategy.

2) Exchange Sequence

The exchange sequence is a fundamental technique used in sequence computation procedures to highlight chromosome differentiation. It allows for the alteration of chromosome sequences without the need for additional procedures, making it a simple and efficient method.

In the context of the DNWFSP, the exchange sequence is a standard heuristic method. Its primary objective is to find an effective plan by rearranging the sequence of two processes on a machine while ensuring compliance with problem constraints. The algorithm aims to discover a better schedule with a shorter completion time by applying the exchange sequence to an initial schedule. It is important to note that the exchange sequence is a heuristic approach and cannot guarantee the best possible outcome. Once an exchange sequence between two chromosomes has been determined in the DNWFSP, it is crucial to implement it promptly. The implementation involves making the necessary changes to the chromosome sequences based on the identified exchange sequence. This immediate implementation allows the algorithm to progress toward generating a potentially more optimal solution for the scheduling problem. While the exchange sequence is a valuable technique, it is vital to understand its heuristic nature and the absence of a guarantee for the best outcome.

3) Genetic operator

A search strategy using natural selection and genetics is a genetic algorithm. It can be used in optimization problems where the objective is to select the optimal solution from all available options. A genetic algorithm uses the three crucial operators of crossover, mutation and selection. Genetic operators can be combined to solve multiple problems and demands, including selection, mutation and crossover. These operators are employed in a genetic algorithm's iteration process to generate new individuals continually, progressively increase population fitness, and finally find the ideal global solution.

a) Mutation Operator

During natural evolution, genes can be affected by various unknown factors, leading to unpredictable changes and resulting in variations from the original genes. This phenomenon is known as variation. In evolutionary algorithms, mutation operators are used to introducing variations by changing a small number of genes on the chromosome, which helps the population converge faster toward an optimal solution [50].

The mutation operator can also be used to explore neighbourhood solutions near excellent individuals, thereby improving the population's diversity when individuals are close to the optimal Pareto front through the crossover operation. However, the mutation rate is usually maintained low value to avoid destroying excellent gene fragments in the individual due to mutation.

In this study, changes are made by using the swap mutation tool. To finish the mutation, the exchange mutation process picks two gene sites at random on the chromosomal structure and swaps the data in these genes [51].

In evolutionary systems, the swap modification is just one of the mutation operators that can be used. It does a good job of keeping the population diverse and keeps good gene pieces from being lost. Switch evolution helps come up with new and different methods that may be closer to the best answer by systematically altering gene sequence at two places. This method can be helpful when looking for an answer in a search area that is complicated and has a lot of dimensions.

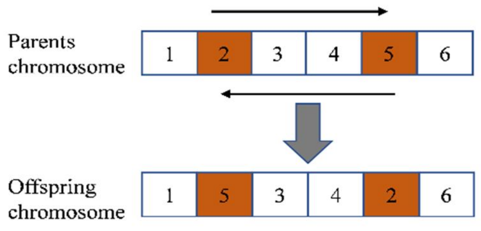

Fig. 4.5 Mutation operator.

Figure 4.5 illustrates an example of the mutation operator in this case, the chromosomes from each parent are 1, 2, 3, 4, 5, and 6. Two gene bits, 2 and 5, are chosen at random for mutation, and the genes on these bits are swapped. This gives the children chromosomes that look like this: 1, 5, 3, 4, 2, 6.

In evolutionary algorithms, mutation operators are used to change the population and look for new answers. In the example, the mutation operator swaps the genes at two bits chosen at random. In this case, bits 2 and 5 are picked, and the genes at places 1 and 6 are switched.

The chromosome that comes out of this, with the numbers 1, 5, 3, 4, 2, is different from the chromosomes of the parents. A quick and efficient technique to introduce changes to a population while maintaining its diversity is to use the swap mutation operator. But it may not always be the best operator for a given situation. Sometimes, other mutation operators may work better.

b) Crossover Operator

The goal of crossing is to make new solutions by taking the best parts of various approaches in the population and putting them together. The crossover operator is based on how sexual reproduction works in nature, where traits from both parents are passed on to the children.

The crossover operator chooses two parent a pair of from the sample and swaps their genetic material to make one or more child chromosomes. There are different kinds of crossover operators, such as one-point crossover, two-point crossover, and uniform crossover. Each kind of crossover operator is made to handle a certain kind of problem and answer representation [52]. For instance (as shown in figure 4.6), the first parental chromosomes in the crossover operation are 1, 2, 3, 4, 5, 6, and the second parental chromosomal is 6, 5, 4, 3, 2, 1. The gene pieces marked by "first" and "second" in the first parent's chromosome are "3" and "4". These gene pieces will remain on the offspring chromosome in the exact location and order. In these gene bits from the second parent, on the other hand, chromosomal regions will no longer be chosen to avoid an illegal answer. Then, the unselected genes from the second parent chromosomal are put into the offspring's chromosomal after the gene marked 2. The process continues from the parent after the initial parent gene is transferred to the offspring. Before choosing all of the genetic material, the first gene on each chromosome is selected [53].

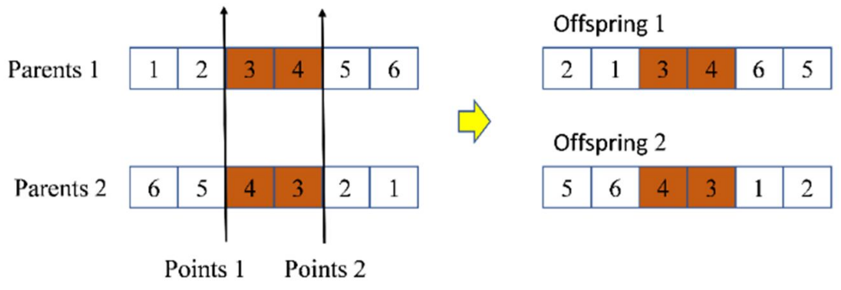

Fig. 4.6 Crossover operator.

c) Selection Operator

The population's top performers are chosen based on the measure of fitness, while the population's worst performers are excluded. In the evolutionary algorithm, such an operation is called a selection operation. This research adopts the binary tournament selection method when using the selection operator. When applying the edge area sampling technique, choose two people at random, compare their function values for one of the optimization targets, choose the person who performs better on the target, and split them into the corresponding subpopulation. At the same time, the central area sampling strategy based on PDDR-FF is used to select individuals with excellent performance in the population. Individuals in the population are categorized as non-dominated based on the lower PDDR-FF score. Choose a select group of people who do the best as the elite group to increase convergence. The binary competitive selection method can protect individuals with high fitness values without losing diversity.

D. Local Search Strategy

The local search strategy known as differential evolution is an algorithm that utilizes the diversity among individuals to guide their evolution, treating each individual in a population as a vector employs differential operations to generate new vectors as evolutionary offspring. Specifically, the difference operation involves creating a new individual vector by adding the difference vector to an existing individual vector, which is determined by computing the difference between three randomly selected individual vectors. This method can be applied in a multi-objective optimization scenario to uncover an optimal global solution.

One of the key advantages of the differential evolution method is its computational efficiency, as it requires only one function evaluation per iteration. This makes it a highly efficient algorithm for solving optimization problems. The precise procedure of the differential evolution approach is illustrated in Figure 4.7.

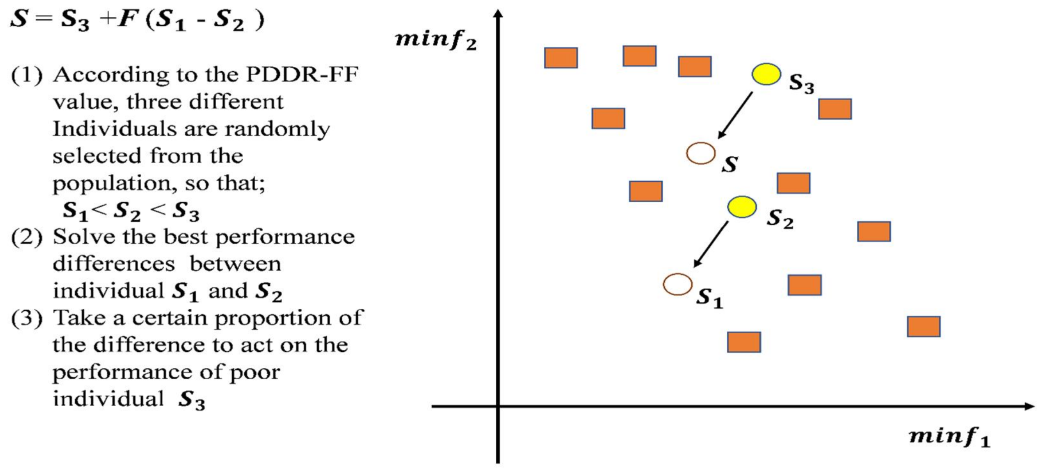
$S = S_3 + F(S_1 - S_2)$
(1) According to the PDDR-FF value, three different Individuals are randomly selected from the population, so that; $S_1 < S_2 < S_3$
(2) Solve the best performance differences between individual S_1 and S_2
(3) Take a certain proportion of the difference to act on the performance of poor individual S_3
$minf_2$
S_3
S
S_2
S_1
$minf_1$

Fig. 4.7 local research strategy.

Variable Neighbourhood Search (VNS) is a powerful metaheuristic technique for combinatorial optimization problems. In recent years, it has acquired popularity due to its ability to solve various combinatorial problems. VNS utilizes a neighbourhood structure that encompasses both inter-machine and intra-machine neighbourhoods. In the context of the DNWFSP, which involves scheduling tasks across multiple machines to minimize the makespan, a VNS method has been proposed [54].

The VNS algorithm consists of two main steps: the construction and search phases. In the construction phase, an initial solution is built using the Nawaz-Enscore-Ham (NEH) heuristic [55], commonly used in flow-shop scheduling problems. The NEH heuristic incrementally adds tasks to the schedule to minimize the total duration. Following the construction phase, the VNS algorithm employs fluctuation operators to generate a set of alternative solutions. These operators include the incorporation and swap operators [56]. The swap operator rearranges the order of two jobs on the same machine, while the insertion operator assigns a task to a different machine.

To enhance the algorithm's performance, local search operators are incorporated into the VNS approach. The algorithm employs the 2-exchange and 3-exchange operators for local search, which involve swapping the positions of two or three jobs [57]. The VNS method was evaluated using various benchmark instances from relevant literature [58]. Experimental results demonstrated that the VNS algorithm outperforms other metaheuristic techniques, such as simulated annealing and tabu search, in solving the DNWFSP.

The VNS algorithm utilizes an inter-machine and intra-machine neighbourhood structure, which forms the basis for its exploration and exploitation capabilities. The experimental findings showcased the competitiveness of the VNS algorithm compared to existing metaheuristic algorithms in solving the DNWFSP and its ability to generate high-quality solutions for large-scale instances [59]. VNS is a metaheuristic optimization approach that systematically explores the neighbourhoods surrounding a given solution in search of the best possible solution. It incorporates a local search within each iteration, and if no improvement is found, it moves to a new neighbourhood to continue the search.

VNS can also be applied to distributed scheduling problems, aiming to optimize the allocation of jobs across resources in a distributed computing system. In such scenarios, multiple tasks must be scheduled across devices with varying processing power and availability. The objective is to minimize the overall makespan, representing the total time required to complete all jobs. The following example illustrates how VNS can be beneficial in distributed scheduling:

1) Start with an initial random allocation of tasks to machines.
2) Define a set of neighbourhoods, each corresponding to a different way of reallocating some tasks to different machines.
3) Conduct a local search for each neighborhood, attempting various combinations of task reallocation inside that neighborhood, until a locally best approach is discovered.
4) Update the global best solution if the locally optimum solution discovered in the current neighborhood is superior than the global best approach.
5) Move to the next neighbourhood and repeat steps 3-4 until all neighbourhoods have been explored.
6) If no improvement is found after exploring all neighbourhoods, restart the algorithm from a new random initial allocation and repeat steps 2-5 until a satisfactory solution is found.

We update the global ideal solution if a locally optimum solution in the immediate area outperforms the present globally solution that is best.
The procedure is repeated until all neighbourhoods have been investigated.
By iteratively exploring different neighbourhoods and performing local searches within each neighbourhood, VNS can efficiently search for optimal solutions to distributed scheduling problems.

V. EXPERIMENT AND RESULT ANALYSIS

In this research, the DNWFSP for Taillard's benchmark set is solved using HEMOEA-DE. This mixed DNWFSP benchmark problem was used. The standard problem has a range of sizes, with n = 20_5, 20_10, 20_20, 50_5, 50_10, 50_20, and 100_5, 100_10, and 100_20 being the number of jobs. Each of the issue sets made up of the benchmark tasks.
The generational distance (*GD*), inverted generational distance (*IGD*), and hyper volume (*HV*) are used to assess the algorithm's efficacy. The suggested method is contrasted with several other algorithms, including the Strength Pareto evolutionary algorithm (SPEA2) the multi-objective evolutionary algorithm based on decomposition (MOEA/D), [60] the non-dominated genetic sorting algorithm II (NSGA-II), and hybrid enhanced multi-objective evolutionary algorithm (HEMOEA). To make the experiment more fair, the average value of the index results produced after each technique is performed 30 times is considered an experiment. The population size is 100 and the algorithm has a maximum generation size of 600. The crossover operator of the comparison algorithm employs sequential crossover, the mutation operation makes use of the exchange mutation operation, and the algorithm's selection operators use the binary tournament selection technique.
To ensure the experiment is fair, all algorithms are also programmed in the Java programming language. The experiments closely follow the guidelines provided in the initial literature. The simulation tests are performed on a single server running Windows Server 2019 and equipped with an Intel Xeon Gold 5218 CPU operating at 2.30 GHz and 64GB of RAM. To find the Pareto set, three comparison measurements *GD, IGD, HV* and *C* are employed.

A. Parameter settings

The comparative experiments were conducted between the HEMOEA-DE, MOHEA, NSGA-II, SPEA2, and MOEA/D. The experimental test problems are DNWFSP based on the FSP benchmark problem proposed. The algorithm's average result was calculated after completing the problems for the experiment's validity. The HEMOEA, SPEA2, NSGA-II and MOEA/D are compared with the HEMOEA-DE method. The parameters of the algorithms are shown in Table 5.1. To evaluate the algorithm's convergence and performance, use *GD, IGD* and *HV* indicators and then analyse the validity of the index results.
All algorithms have a predetermined population size and a maximum number of iterations. Because only NSGA-II, SPEA2, MOEA/D, HEMOEA and HEMOEA-DE have crossover and mutation operations, their crossover rate and mutation rate are set to the same value, as is the size of the elite group. Two subgroups of the HEMOEA-DE algorithm are required. Set the size of the two subgroups to the population's size, and the sum matches the population's size.

Table 5. 1 The algorithm parameter setting.

Parameters	HEMOEA-DE	HEMOEA	NSGA-II	SPEA2	MOEA/D
Population Size	100	100	100	100	100
Elite Population Size	50	50	\	50	\
Sub-population Size	50	50	\	\	\
Crossover Rate	0.7	0.7	0.7	0.7	0.7
Mutation Rate	0.3	0.3	0.3	0.3	0.3
DE proportion	0.7	\	\	\	\
Maximum Generation	600	600	600	600	600

B. Experimental Results and Analysis

Table 5.2 The *GD* index results.

GD	HEMOEA-DE	HEMOEA	NSGA-II	SPEA2	MOEA/D
20_5	1.07E-01	1.22E-01	1.87E-01	1.65E-01	1.83E-01
20_10	3.93E-02	5.22E-02	7.61E-02	9.66E-02	1.13E-01
20_20	8.03E-02	1.16E-01	1.31E-01	1.29E-01	1.36E-01
50_5	1.27E-01	1.30E-01	1.55E-01	1.47E-01	1.60E-01
50_10	6.91E-02	7.51E-02	9.61E-02	8.37E-02	9.59E-02
50_20	5.93E-02	7.33E-02	8.65E-02	9.55E-02	7.36E-02
100_5	8.60E-02	8.12E-02	1.24E-01	1.16E-01	1.11E-01
100_10	4.61E-02	6.41E-02	8.09E-02	7.71E-02	6.75E-02
100_20	4.12E-02	4.49E-02	5.84E-02	6.01E-02	6.57E-02

Table 5.2 displays the *GD* index results. The Pareto front can be approached, as shown by the *GD* index. The more closely the solution gets to the Pareto front, the lower the *GD*. The HEMOEA-DE algorithm performed best on nine tasks, while SPEA2 performed well on problems. The proposed method exhibits no significant divergence from the performance of NSGA-II and HEMOEA. Specifically, on a particular problem, it demonstrates no notable deviation from the outcomes achieved by HEMOEA/D. In all other settings, HEMOEA-DE outperforms all competing algorithms. This indicates that the solution discovered by HEMOEA-DE is closer to the Pareto frontier. HEMOEA-DE has clear advantages regarding convergence over the comparison algorithm, which suggests that HEMOEA-DE is better at convergence.

Table 5. 3 The *HV* index results.

HV	HEMOEA-DE	HEMOEA	NSGA-II	SPEA2	MOEA/D
20_5	1.84E+05	1.72E+05	8.66E+04	1.14E+05	9.13E+04
20_10	4.39E+05	4.18E+05	3.21E+05	2.97E+05	2.44E+05
20_20	8.84E+05	6.57E+05	4.88E+05	4.97E+05	4.69E+05
50_5	6.92E+05	6.66E+05	4.96E+05	5.47E+05	4.59E+05
50_10	9.78E+05	9.12E+05	6.29E+05	7.68E+05	6.63E+05
50_20	2.29E+06	1.88E+06	1.44E+06	1.20E+06	1.95E+06
100_5	1.85E+06	1.86E+06	6.45E+05	8.34E+05	1.01E+06
100_10	6.39E+06	4.23E+06	3.22E+06	3.30E+06	4.21E+06
100_20	1.32E+07	1.22E+07	9.68E+06	9.48E+06	8.28E+06

Table 5.4 The *IGD* results.

IGD	HEMOEA-DE	HEMOEA	NSGA-II	SPEA2	MOEA/D
20_5	9.45E-02	1.05E-01	1.84E-01	1.54E-01	1.80E-01
20_10	3.34E-02	4.56E-02	6.89E-02	8.01E-02	1.06E-01
20_20	7.43E-02	1.03E-01	1.25E-01	1.23E-01	1.31E-01
50_5	1.24E-01	1.28E-01	1.51E-01	1.45E-01	1.58E-01
50_10	6.57E-02	7.17E-02	9.16E-02	8.11E-02	9.38E-02
50_20	5.86E-02	7.40E-02	8.51E-02	9.42E-02	7.45E-02
100_5	8.53E-02	8.07E-02	1.23E-01	1.16E-01	1.12E-01
100_10	4.71E-02	6.63E-02	8.18E-02	8.00E-02	7.04E-02
100_20	4.96E-02	5.35E-02	6.55E-02	6.65E-02	7.29E-02

Results for every *HV* index method are shown in Table 5.3. The *HV* index may exemplify the method's convergence and distributed performance. The algorithm’s convergence and distribution performance improve. Similarly, HEMOEA-DE scores best on 8 of 9 test issues, while HEMOEA performs best on 1 problem. This showed that the proposed method, HEMOEA-DE, performs better in convergence and distribution than competing methods.
The output of each method on the *IGD* index is shown in Table 4.4. The *IGD* index, like the *HV* index, can evaluate the algorithm's convergence and distributed performance simultaneously. The efficiency of the method improves with decreasing *IGD* index. Likewise, out of 9 test problems, HEMOEA-DE performs best on 8 problems, while HEMOEA performs best on 1 problem. This demonstrates how the suggested method, HEMOEA-DE, performs better in terms of convergence and distribution.
In conclusion, the findings of the *GD*, *IGD*, *HV*, and *C* indexes show that HEMOEA-DE has greater convergence and performance. As a result, the DE can improve the algorithm’s convergence performance while ensuring highly distributed performance.

Table 5. 5 The *C* index results of HEMOEA-DE.

	A: HEMOEA-DE B: HEMOEA D: NSGA-II E: SPEA2 F: MOEA/D							
C	C (A, B)	C (B, A)	C (A, D)	C (D, A)	C (A, E)	C (E, A)	C (A, F)	C (F, A)
20_5	4.33E-01	3.67E-01	8.33E-01	6.67E-02	7.67E-01	0.00E+00	7.33E-01	6.67E-02
20_10	3.00E-01	2.00E-01	6.67E-01	1.00E-01	7.33E-01	1.00E-01	7.67E-01	3.33E-02
20_20	6.67E-01	1.00E-01	9.00E-01	0.00E+00	7.67E-01	3.33E-02	7.67E-01	1.00E-01
50_5	3.33E-01	4.00E-01	5.33E-01	3.00E-01	5.33E-01	1.67E-01	6.67E-01	1.33E-01
50_10	4.33E-01	3.00E-01	5.67E-01	1.33E-01	4.00E-01	3.00E-01	5.33E-01	2.00E-01
50_20	5.33E-01	1.67E-01	6.67E-01	1.00E-01	7.33E-01	6.67E-02	5.33E-01	2.67E-01
100_5	3.67E-01	4.67E-01	7.33E-01	1.33E-01	6.00E-01	2.33E-01	5.67E-01	2.33E-01
100_10	6.67E-01	1.00E-01	7.33E-01	1.00E-01	7.00E-01	1.00E-01	5.33E-01	1.00E-01
100_20	3.67E-01	3.33E-01	5.00E-01	1.67E-01	6.67E-01	2.67E-01	6.33E-01	1.33E-01

The algorithm's converging effectiveness is assessed using the *C* index. With an increase in the index, the algorithm approaches the Pareto boundary. The *C* index values in Table 5.5 reveal that HEMOEA-DE is superior to NSGA-II, SPEA2, MOEA/D, and HEMOEA. On seven problems, HEMOEA-DE outperforms HEMOEA, while on two problems, HEMOEA outperforms HEMOEA. HEMOEA-DE has the best convergence performance, as indicated by the *C* index. The HEMOEA-DE shows obvious advantages based on *IGD* indicators, showing that the HEMOEA-DE has the best convergence performance. The *HV* indications show that HEMOEA-DE is superior to other algorithms and that HEMOEA-DE has a greater breadth of solutions. The *C* index further demonstrates that HEMOEA-DE has the best convergence performance. The convergence performance of HEMOEA-DE is significantly superior, and the search solutions’ performance is adequate.
The DNWFSP is a critical problem that requires an optimized solution. This paper suggests using a spreadsheet environment to apply the HEMOEA-DE optimization approach to tackle this challenge. VEGA's and the new PDDR-FF-based sampling techniques are combined in HEMOEA's hybrid sampling strategy. The HEMOEA-DE algorithm is developed by combining DE and HEMOEA based on this hybrid sampling strategy. Additional factories are added to the benchmark test problems of DNWFSP to conduct experiments. The HEMOEA-DE algorithm is tested on the benchmark problem put out by E. Taillard, and it is evaluated in comparison with various other methods, including MOEA/D, SPEA2, NSGA-II, and HEMOEA. According to the experiments conducted, the method suggested in this paper outperforms the comparison algorithms while solving the DNWFSP benchmark issues. The HEMOEA-DE algorithm is characterized by its fast convergence and excellent performance.

VI. CONCLUSION

This paper focuses on the DNWFSP in the context of the manufacturing industry. The paper aims to develop a suitable mathematical formulation for DNWFSP and design an encoding and decoding method using a chromosome representation. A genetic operator is designed to facilitate chromosome recombination while avoiding infeasible solutions. The HEMOEA algorithm accelerates convergence towards the Pareto frontier and enhances solution performance. Additionally, a local search algorithm based on the difference in chromosome sequence, inspired by DE, is incorporated. This strengthens the algorithm's ability to perform local searches and improves the quality of the chromosomes.

Experiments are conducted using the DNWFSP benchmark problem proposed by E. Taillard, adding variables representing the number of factories. Multiple indicators are used to evaluate the convergence and performance of the HEMOEA-DE algorithm.

REFERENCES

[1] Pezzella, Ferdinando, Gianluca Morganti, and Giampiero Ciaschetti. "A genetic algorithm for the flexible job-shop scheduling problem." Computers & operations research 35.10 (2008): 3202-3212
[2] Shao, Zhongshi, Dechang Pi, and Weishi Shao. "Hybrid enhanced discrete fruit fly optimization algorithm for scheduling blocking flow-shop in distributed environment." Expert Systems with Applications 145 (2020): 113147.
[3] Behnamian, Javad, and S. M. T. Fatemi Ghomi. "A survey of multi-factory scheduling." Journal of Intelligent Manufacturing 27 (2016): 231-249.
[4] Yazdani, Mehdi, Sheida Gohari, and Bahman Naderi. "Multi-factory parallel machine problems: Improved mathematical models and artificial bee colony algorithm." Computers & Industrial Engineering 81 (2015): 36-45.
[5] Lu, Chao, et al. "A Pareto-based collaborative multi-objective optimization algorithm for energy-efficient scheduling of distributed permutation flow-shop with limited buffers." Robotics and Computer-Integrated Manufacturing 74 (2022): 102277.
[6] Meng, Tao, Quan-Ke Pan, and Ling Wang. "A distributed permutation flowshop scheduling problem with the customer order constraint." Knowledge-Based Systems 184 (2019): 104894.
[7] Hatami, Sara, Ruben Ruiz, and Carlos Andres-Romano. "The distributed assembly permutation flowshop scheduling problem." International Journal of Production Research 51.17 (2013): 5292-5308.
[8] Xiong, Fuli, et al. "Just-in-time scheduling for a distributed concrete precast flow shop system." Computers & Operations Research 129 (2021): 105204.
[9] Gen, Mitsuo, Runwei Cheng, and Lin Lin. Network models and optimization: Multiobjective genetic algorithm approach. Springer Science & Business Media, 2008.
[10] Jiao, Bin, and Shaobin Yan. "A niche sharing scheme-based co-evolutionary particle swarm optimization algorithm for flow shop scheduling problem." Syst. Cybernet. Inf 15 (2017): 46-54.
[11] Yu, Xinjie, and Mitsuo Gen. Introduction to evolutionary algorithms. Springer Science & Business Media, 2010.
[12] Gao, Jian, Rong Chen, and Wu Deng. "An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem." International Journal of Production Research 51.3 (2013): 641-651.
[13] Wang, Sheng-yao, et al. "An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem." International Journal of Production Economics 145.1 (2013): 387-396.
[14] Xu, Ye, et al. "An effective hybrid immune algorithm for solving the distributed permutation flow-shop scheduling problem." Engineering Optimization 46.9 (2014): 1269-1283.
[15] Naderi, Bahman, and Rubén Ruiz. "A scatter search algorithm for the distributed permutation flowshop scheduling problem." European Journal of Operational Research 239.2 (2014): 323-334.
[16] Bargaoui, Hafewa, Olfa Belkahla Driss, and Khaled Ghédira. "A novel chemical reaction optimization for the distributed permutation flowshop scheduling problem with makespan criterion." Computers & Industrial Engineering 111 (2017): 239-250.
[17] Zhao, Fuqing, et al. "An optimal block knowledge driven backtracking search algorithm for distributed assembly No-wait flow shop scheduling problem." Applied Soft Computing 112 (2021): 107750.
[18] Kaveh, Ali, and Taha Bakhshpoori. "Metaheuristics: outlines, MATLAB codes and examples." (2019): 978-3.
[19] Zhao, Fuqing, et al. "A discrete water wave optimization algorithm for no-wait flow shop scheduling problem." Expert Systems with Applications 91 (2018): 347-363.
[20] Chen, Jing-fang, Ling Wang, and Zhi-ping Peng. "A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling." Swarm and Evolutionary Computation 50 (2019): 100557.
[21] Fernandez-Viagas, Victor, Paz Perez-Gonzalez, and Jose M. Framinan. "The distributed permutation flow shop to minimise the total flowtime." Computers & Industrial Engineering 118 (2018): 464-477.
[22] Liao, Zhifang, et al. "GRBMC: An effective crowdsourcing recommendation for workers groups." Expert Systems with Applications 179 (2021): 115039.
[23] Amin Naseri, M. R., Saeed Tasouji Hassanpour, and N. Nahavandi. "Solving re-entrant no-wait flow shop scheduling problem." International Journal of Engineering 28.6 (2015): 903-912.
[24] Ishibuchi, Hisao, Tadashi Yoshida, and Tadahiko Murata. "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling." IEEE transactions on evolutionary computation 7.2 (2003): 204-223.
[25] Zhao, Fuqing, et al. "A two-stage differential biogeography-based optimization algorithm and its performance analysis." Expert Systems with Applications 115 (2019): 329-345.
[26] Engin, Orhan, and Abdullah Güçlü. "A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems." Applied Soft Computing 72 (2018): 166-176.
[27] Pan, Quan-Ke, Ling Wang, and Bin Qian. "A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems." Computers & Operations Research 36.8 (2009): 2498-2511.
[28] Zheng, Tianmin, and Mitsuo Yamashiro. "Solving no-wait flow shop scheduling problems by a hybrid quantum-inspired evolutionary algorithm." Advances in Soft Computing: 9th Mexican International Conference on Artificial Intelligence, MICAI 2010, Pachuca, Mexico, November 8-13, 2010, Proceedings, Part II 9. Springer Berlin Heidelberg, 2010.
[29] Chaudhry, Imran A., Riaz Ahmed, and Abdul Munem Khan. "Genetic Algorithm to minimize flowtime in a no-wait flowshop scheduling problem." IOP Conference Series: Materials Science and Engineering. Vol. 65. No. 1. IOP Publishing, 2014.
[30] Qu, Chiwen, et al. "Solutions to no-wait flow shop scheduling problem using the flower pollination algorithm based on the hormone modulation mechanism." Complexity 2018 (2018).
[31] Idertsogt, Sainzaya, and Wenqiang Zhang. "Hybrid multi-objective evolutionary algorithm with differential evolution for no-wait flowshop scheduling

problem." (2020).
[32] Jiang, En-da, and Ling Wang. "An improved multi-objective evolutionary algorithm based on decomposition for energy-efficient permutation flow shop scheduling problem with sequence-dependent setup time." International Journal of Production Research 57.6 (2019): 1756-1771.
[33] Cheng, Tai C. Edwin, Bo Peng, and Zhipeng Lü. "A hybrid evolutionary algorithm to solve the job shop scheduling problem." Annals of Operations Research 242.2 (2016): 223-237.
[34] Habibi, Sara, Shahin Ordikhani, and Ahmad Reza Haghighi. "Solving Re-entrant No-wait Flexible Flowshop Scheduling Problem; Using the Bottleneck-based Heuristic and Genetic Algorithm." Journal of Modern Processes in Manufacturing and Production 7.2 (2018): 65-77.
[35] Shao, Weishi, Dechang Pi, and Zhongshi Shao. "An extended teaching-learning based optimization algorithm for solving no-wait flow shop scheduling problem." Applied Soft Computing 61 (2017): 193-210.
[36] Bewoor, Laxmi A., V. Chandra Prakash, and Sagar U. Sapkal. "Evolutionary hybrid particle swarm optimization algorithm for solving NP-hard no-wait flow shop scheduling problems." Algorithms 10.4 (2017): 121.
[37] Wang, Xianpeng, et al. "Multiobjective multitask optimization-neighborhood as a bridge for knowledge transfer." IEEE Transactions on Evolutionary Computation (2022).
[38] Chang, Yu-Teng, and Tsung-Che Chiang. "Multiobjective permutation flow shop scheduling using MOEA/D with local search." 2016 Conference on Technologies and Applications of Artificial Intelligence (TAAI). IEEE, 2016.
[39] Wang, Hongfeng, et al. "A NSGA-II based memetic algorithm for multiobjective parallel flowshop scheduling problem." Computers & Industrial Engineering 113 (2017): 185-194.
[40] Nouri, Nouha, and Talel Ladhari. "Evolutionary multiobjective optimization for the multi-machine flow shop scheduling problem under blocking." Annals of Operations Research 267 (2018): 413-430.
[41] Wei, Xiu, et al. "Multi-objective local search combined with NSGA-II for Bi-criteria permutation flow shop scheduling problem." IEEJ Transactions on Electronics, Information and Systems 132.1 (2012): 32-41.
[42] Zhang, Wenqiang, et al. "Fast multi-objective hybrid evolutionary algorithm for flow shop scheduling problem." Proceedings of the Tenth International Conference on Management Science and Engineering Management. Springer Singapore, 2017.
[43] Zhang, Wenqiang, Mitsuo Gen, and Jungbok Jo. "Hybrid sampling strategy-based multiobjective evolutionary algorithm for process planning and scheduling problem." Journal of Intelligent Manufacturing 25 (2014): 881-897.
[44] Zhang, Wenqiang, et al. "Hybrid evolutionary algorithm with sequence difference-based differential evolution for multi-objective fuzzy flow-shop scheduling problem." International Journal of Internet Manufacturing and Services 8.4 (2022): 308-329.
[45] Schaffer, J. David. "Multiple objective optimizations with vector evaluated genetic algorithms." Proceedings of the first international conference on genetic algorithms and their applications. Psychology Press, 2014.
[46] Zhang, Wenqiang, et al. "Fast multiobjective hybrid evolutionary algorithm based on mixed sampling strategy." Proceedings of the Eleventh International Conference on Management Science and Engineering Management 11. Springer International Publishing, 2018.
[47] Bhaskar, V., Santosh K. Gupta, and Ajay K. Ray. "Applications of multiobjective optimization in chemical engineering." Reviews in chemical engineering 16.1 (2000): 1-54.
[48] Zhang, Qingfu, and Hui Li. "MOEA/D: A multiobjective evolutionary algorithm based on decomposition." IEEE Transactions on evolutionary computation 11.6 (2007): 712-731.
[49] Fichera, Sergio, Antonio Costa, and Fulvio Cappadonna. "Scheduling jobs families with learning effect on the setup." Advances in Operations Research 2015 (2015).
[50] Shao, Weishi, Dechang Pi, and Zhongshi Shao. "Optimization of makespan for the distributed no-wait flow shop scheduling problem with iterated greedy algorithms." Knowledge-Based Systems 137 (2017): 163-181.
[51] Zhang, Qingfu, and Hui Li. "MOEA/D: A multiobjective evolutionary algorithm based on decomposition." IEEE Transactions on evolutionary computation 11.6 (2007): 712-731.
[52] Wu, Xiangjuan, et al. "A new hybrid algorithm for solving large scale global optimization problems." IEEE Access 7 (2019): 103354-103364.
[53] Shao, Weishi, Zhongshi Shao, and Dechang Pi. "Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem." Knowledge-Based Systems 194 (2020): 105527.
[54] Sastry, Kumara, David Goldberg, and Graham Kendall. "Genetic algorithms." Search methodologies: Introductory tutorials in optimization and decision support techniques (2005): 97-125.
[55] Wang, Xiaojuan, et al. "A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem." The International Journal of Advanced Manufacturing Technology 51.5-8 (2010): 757-767.
[56] Panigrahi, Bijaya Ketan, et al., eds. Swarm, Evolutionary, and Memetic Computing: 4th International Conference, SEMCCO 2013, Chennai, India, December 19-21, 2013, Proceedings, Part I. Vol. 8297. Springer, 2013.
[57] Gao, Liang, et al. "A hybrid algorithm based on a new neighborhood structure evaluation method for job shop scheduling problem." Computers & Industrial Engineering 88 (2015): 417-429.
[58] Bouffard, Véronique, and Jacques A. Ferland. "Improving simulated annealing with variable neighborhood search to solve the resource-constrained scheduling problem." Journal of Scheduling 10.6 (2007): 375-386.
[59] Hansen, Pierre, Nenad Mladenović, and Dionisio Perez-Britos. "Variable neighborhood decomposition search." Journal of Heuristics 7.4 (2001): 335-350.
[60] Taillard, Eric. "Benchmarks for basic scheduling problems." european journal of operational research 64.2 (1993): 278-285.