International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(21s) 2026 pp. 131-158
https://doi.org/10.70917/ijcisim-2026-4611
Article

An estimation of distribution algorithm based on the Dirichlet distribution for the flexible job shop scheduling problem

Ricardo Pérez-Rodríguez[1], María de los Ángeles Silva-Olvera[2], Sergio Frausto-Hernández[3], José Francisco Louvier-Hernández[3]

[1]SECIHTI – TECNM Aguascalientes Institute of Technology, Economic and Administrative Department, Adolfo López Mateos 1801 Ote, Bona Gens, CP 20255 Aguascalientes city, Mexico.
[2]TECNM Aguascalientes Institute of Technology, Economic and Administrative Department, Adolfo López Mateos 1801 Ote, Bona Gens, CP 20255 Aguascalientes city, Mexico.
[3]TECNM Aguascalientes Institute of Technology, Chemical and Biochemical Engineering Department, Adolfo López Mateos 1801 Ote, Bona Gens, CP 20255 Aguascalientes city, Mexico.
Corresponding author
email: dr.ricardo.perez.rodriguez@gmail.com

Abstract: This article proposes a novel Estimation of Distribution Algorithm (EDA), named DEDA, which leverages the Dirichlet distribution to address the Flexible Job Shop Scheduling Problem (FJSSP). The FJSSP is a complex combinatorial optimization problem in which jobs consist of multiple operations processed on various feasible machines under sequence constraints. Complexity of FJSSP motivates the development of efficient algorithms. This research outlines the problem, provides a comprehensive review of diverse methods applied to the FJSSP, and offers an in-depth discussion of solution representations, local search strategies, dynamic environment handling, and multi-objective approaches. Evaluation metrics and benchmarking datasets used in the literature are also summarized. The study introduces the DEDA method, detailing solution representation, population generation, offspring creation using the Dirichlet distribution, fitness evaluation, and population replacement. The manuscript discusses the concept of EDAs, their advantages in building probabilistic models to generate improved offspring, and various modeling approaches, including Bayesian networks, ranking models, and radial distributions. It highlights a gap in the literature regarding EDAs that utilize Dirichlet distributions to enhance offspring quality for combinatorial problems such as the FJSSP. DEDA is tested and compared on standard benchmark datasets against prominent EDAs and metaheuristics, showing statistically significant superior performance in multiple tests. The article concludes that DEDA offers a promising approach for the FJSSP and can be extended to other scheduling and combinatorial optimization problems.
Keywords: Dirichlet distribution; gamma distribution; estimation of distribution algorithms; scheduling; flexible job shop

1. Introduction

The supply chain encompasses that cover the entire lifecycle of a product or service, from conception to consumption [1]. Various participants are involved in any supply chain, including producers, marketers, distributors, retail sellers, as well as elements such as data, raw materials, operators, and capital. The operations within a supply chain include purchasing, manufacturing, transportation, and more. Supply chain management involves a variety of decisions and transactions among the different entities involved, typically categorized as strategic, tactical, and operational [2]. The main activities of the supply chain generally take place within a facility. Raw materials or parts are stored until they are shipped. Within any facility, decisions occur at multiple levels: strategic, mid-term, and operational. Examples include a) Strategic decisions, such as selecting the factory location, designing the layout,

choosing suppliers and carriers, and implementing software systems. b) Mid-term decisions, such as establishing policies to meet quality and safety standards, determining storage locations within the layout, defining the flow of parts in the warehouse, and conducting cyclical inventory operations. c) Operational decisions, such as forecasting demand, scheduling, dispatching receipts and shipments, and managing order arrivals. For further details see references [2-4].

One of the key operational decisions is to determine the optimal scheduling to enhance the overall performance of resources within a facility. Different configurations arise in various manufacturing schemes. This paper analyzes the configuration known as the Flexible Job Shop Scheduling Problem (FJSSP). In the FJSSP, each operation of a job can be processed on any machine within a set of feasible machines. The FJSSP is notoriously difficult [5]. While the classical Job Shop Scheduling Problem (JSSP) focuses on finding the optimal sequence of operations, the FJSSP also addresses the machine assignment decision.

Given the combinatorial nature of FJSSP, one active area of research involves combining different methods to develop more robust algorithms [7]. In this study, we continue this line of research by incorporating a new estimation of distribution algorithm (EDA). The primary goal of research on EDAs is to identify improved methods for constructing probability models to enhance their performance. Accordingly, we introduce the use of the Dirichlet distribution to improve the performance of an EDA, which we call DEDA, to effectively solve the FJSSP.

Dirichlet distribution has been applied in various contexts. For example, in topic modeling, it enables the discovery of hidden themes within a corpus of documents by modeling the distribution of themes per document and words per theme. It is also used to model user preferences, such as distributions over themes, thereby enhancing content personalization. Additionally, it is useful in mixture models, image segmentation, and dimensionality reduction, where probabilistic proportions across multiple categories are required. In genomics, the Dirichlet distribution is applied to analyze ribonucleic acid sequencing counts, facilitating the study of gene isoform usage. In summary, the Dirichlet distribution is fundamental when data represents proportions, relative frequencies, or probabilities that must sum to one, allowing for robust handling of uncertainty.

1.1 Problem statement

Based on [6] the FJSSP can be understood as follows:

We have a set of n jobs $J = \{J_1, \ldots, J_n\}$ and a set of m machines $M = \{M_1, \ldots, M_m\}$. Each job J consists of a sequence of O operations, that is to say, $J_i = \{O_{i,1}, \ldots, O_{i,n_i}\}$ where n_i is the number of operations for the job i. Each operation $O_{i,j}$ can be processed on some machine within a set of feasible machines $M_{ij} \subseteq M$. In the FJSSP, the following conditions must be taken into consideration

- At the start of the planning horizon, all data is available in advance, including jobs, operations, and feasible machines.
- Each job has a predetermined sequence of operations.
- The operations for each job must be processed sequentially.
- There is no relationship between the jobs.
- The machines can process one operation at a time.
- The operation cannot be interrupted once the process has started on the assigned machine.
- Breakdowns, setup times, and transport times are not taken into account.
- All processing times cannot be negative.

2. Relevant literature review

Over the past decades, numerous studies have applied meta-heuristic algorithms to solve the FJSSP. [8] utilized dispatching rules to initialize a population of feasible solutions. Each feasible solution consists of two parts: an operation sequence representation as the first part and a machine selection representation as the second part. In the first part, the following dispatching rules were applied: the most work remaining rule, the random rule, and the greatest number of operations remaining rule. In the second part, the random rule, operation minimum processing time rule, and local minimum processing time rule were used. The initial population was generated using a harmony search (HS) algorithm that incorporates these rules. Subsequently, an Artificial Bee Colony (ABC) algorithm was implemented to generate new feasible solutions (also called neighboring solutions) through the Iterated Local Search (ILS) and Scatter Search (SS) techniques. One of these local search techniques is selected using the roulette wheel selection scheme. Next, a tournament selection step is performing among the solutions. If a selected solution is a local optimum, a Simulated Annealing (SA) algorithm is applied to escape from the current position. Otherwise, a Filter and Fan (F&F) algorithm is utilized for the selected solution. Following this, the search is enhanced by a crossover operation. For the operation sequence part, the authors employed a Modified Precedence Operation Crossover (MPOX) operator. For the machine assignment part, a uniform crossover operator was used. Finally, the best parents and the best offspring form the population for the next generation. The proposed ABC algorithm was tested using the dataset from [9], which includes 10 instances: MK01 – MK10. The results of the proposed algorithm were compared to those of the Pareto-based discrete harmony search algorithm discussed in [10], the biogeography-based optimization (BBO) algorithm detailed in [11], and a modified simulated annealing method implemented in [12].

The authors concluded that, considering all 10 instances, the metric (average percentage deviation) of the proposed ABC scheme outperforms the compared methods.

[13] also presented an ABC algorithm. The authors utilized a combination of two types of vectors, as in [8]'s research: the operation sequence vector and the machine assignment vector, which we refer to as type 1 and type 2 vectors respectively. The initial population was generated using a hybrid approach. First, type 1 vectors were initialized using three rules: the random rule, the most work remaining rule (MWR), and the greatest number of operations remaining rule (MOR). Meanwhile, type 2vectors were initialized using three different rules: the random rule, the local minimum processing time rule, and the global minimum processing time rule. Additionally, three crossover operators were applied to evolve type 2 vectors: the two-point crossover, the uniform crossover, and the Precedence Preserving Order Based Crossover (POX). To ensure exploration capability, a mutation operator for type 2 vectors was also included in the proposed scheme. Furthermore, a local search strategy based on the critical path was developed and integrated into the ABC algorithm. The authors also addressed dynamic issues in the manufacturing environment, such as jobs cancellations, jobs arrivals throughout the scheduling, and cancellations of operations within some jobs. To handle these scenarios, they devised heuristics and incorporated them into the ABC algorithm to solve dynamic scheduling problems, by rearranging operations. The performance of their scheme was validated using the dataset from [9], which includes 10 instances: MK01 – MK10. The results of the proposed algorithm were compared with those from [14], [15], [16] (a hybrid tabu search algorithm), [17] (an ABC algorithm), [18] (ant colony optimization), and [19] (a parallel variable neighborhood search algorithm). The authors concluded that the strong performance of the proposed scheme is attributed to the combination of diverse initial populations, the moving operations technique, and the local search method.

[20] presented a hybrid algorithm based on the ant colony system and local search to solve the FJSSP. A key feature of this approach is the integration of scheduling and vehicle routing, with the vehicle routing component solved optimally using mathematical programming. To the best of their knowledge, this is the first reported work in the literature to address scheduling combined with vehicle routing. The combination of a hybrid metaheuristic and a mathematical programming fall under the category of matheuristic algorithms. Their algorithm consists of two phases. The first phase is based on the ant colony system algorithm detailed by [21]. In this phase, each ant constructs a feasible solution in parallel. By tracking artificial pheromones, each ant defines machine production routes. Once all ants have completed their solutions, the best-performing ant proceeds to the second phase, which employs a hill descent algorithm. The implementation of this algorithm is based on the work of [22], which focuses on moving

operations belonging to the critical route of batches that influence the optimization criterion. During the vehicle routing process, the algorithm assigns production orders to vehicles. A mixed-integer programming model is used to determine the shortest path for each trip, thereby minimizing travel time. Experiments were conducted by adapting instances from [23] for the flexible job shop problem. For large instances, the scheme achieves an efficiency improvement between 8% and 12%. Finally, it was concluded that the total completion time in the job shop environment accounts for between 50% and 75.5% of the overall process time.

[24] studied an extension of the FJSSP, known as the Distributed and Flexible Job Shop Scheduling Problem (DFJSSP). In this problem, a set of geographically distributed factories, each containing m machines, processes n jobs. In a globalized market, it is common to have diverse facilities. Businesses benefit from being close to their clients, enabling rapid responses to market shifts. They also experience significant productivity gains due to factors such as variations in labor costs and the skill levels of employees in specific regions. However, the DFJSSP is a complex and intricate problem, similar to other scheduling challenges. It simultaneously considers assigning jobs to machines and distributing jobs across different factories. The authors propose a Chemical Reaction Optimization (CRO) scheme to address the DFJSSP. Their primary objective was to identify the optimal assignment of jobs to factories, followed by building production scheduling within each factory. They developed an initial heuristic that schedules each job in every factory, then selects the factory with the best schedule to assign the job definitively. This process is repeated until all jobs are assigned to factories. Subsequently, diversification and intensification techniques were applied to ensure exploration and exploitation of the search space, respectively. The proposed CRO scheme was evaluated using classical FJSSP instances from the literature, adapted for the DFJSSP from [23], [9], and [25]. The CRO's performance was compared against the Improved Genetic Algorithm (IGA) from [26] and the Genetic Algorithm with a concise chromosome representation (GA-JS) from [27], both designed for the DFJSSP. The results demonstrate that the CRO approach is competitive.

[28] presented various improvements to the cuckoo search (CS) algorithm to address different optimization problems. Consequently, the authors proposed two enhancements for the CS algorithm to solve the FJSSP: Best Neighbors Generation (CS-BNG), and Iterative Levy Flight (CS-ILF). The first method, CS-BNG, improves the CS algorithm during the phase of generating new solutions to replace discarded ones. The authors suggested dividing a portion of abandoned nests into two segments. The first segment is filled by randomly generating solutions, while the second segment is generated randomly based on the neighborhood of the current optimal solution. The second method, CS-ILF, strengthens the CS algorithm's ability to explore new locations within the search space by creating new solutions based on existing ones. The Levy flight step is repeated multiple times during each iteration. Although the paper does not provide an in-depth explanation of the solution representation, it details modifications to the Levy flight equation to better suit discrete spaces. Additionally, the operations involved in the Levy flight process– subtraction, multiplication, and addition– are replaced by neighborhood-based operations designed specifically for the FJSSP. The proposed scheme was applied to the instances. from [23]. Experimental results indicated that both enhancements yield superior solutions and improve the convergence rate in most cases compared to the standard CS algorithm.

[29] presented a simulation-based CS optimization algorithm for the FJSSP. In this study, the fitness value was calculated using discrete-event simulation. A simulation model was developed on the Promodel© simulation platform to accurately represent the shop floor environment. This simulation model was also used to compute performance metrics, as it provides results that closely approximate those of a real system compared to purely mathematical functions. The proposed approach was tested on a sample production order consisting of three jobs and four machines. Additionally, the simulation model accounted for the time each job required to travel between different locations, including machine transfers and movements from incoming/outgoing locations to machines. Furthermore, two case studies from [9] were analyzed. For comparison purposes, the transfer time and throughput were excluded from the simulation process. The results demonstrate that the proposed method effectively addresses the FJSSP for small-sized problems, considering transfer times and throughput.

[30] presents detailed hybrid differential evolution (HDE) algorithms for solving the FJSSP. This research aligns with other studies that utilize metaheuristics for the FJSSP, aiming to discover near-optimal schedules within reasonable computational time, a focus that has intensified over recent decades. The authors generate the initial population randomly, initializing each value in every vector using a uniform random function. To evaluate a chromosome, it is first converted into a discrete two-vector code through a process called forward conversion; then, this two-vector code is decoded into an active schedule [31]. In this study, each chromosome is divided into two separate parts. The first part contains information about machine assignments for each operation, while the second part represents the sequencing of operations on all machines. The machine assignment vector is an array of integer values, where each value indicates that operation i is assigned to the $m\text{-}th$ machine in its alternative machine set. The operation sequence vector is a permutation of all operations. It is important to note that not every permutation is feasible, as some do not preserve priority order within the vector. Therefore, a fixing process is applied to adjust the sequence of operations within each job. The manuscript details the mutation, crossover, and selection processes. Additionally, a well-developed local search algorithm is employed to enhance exploitation for local solution improvement. The authors test their approach on four sets of well-known benchmark instances for the FJSSP and compare their results with those of several algorithms from the literature.

[32] proposed an improved Differential Evolution (DE) algorithm for the FJSSP. The initial population generated with random values between 0 and 1, where the dimension corresponds to the number of operations, and the population size determines the number of target vectors in the sample. After applying mutation and crossover operators, the fitness function ranks the individuals in ascending order. Each operation is assigned according to the ordered vector, and the machine that completes the task the fastest is selected. Additionally, the authors incorporated Local Search (LS) to enhance the search process. The performance of the DE algorithm was validated using benchmark instances from [33] (K01–K05), [9] (MK01–MK10), and [34] (01, 04, 07, 09, and 11).

[35] developed a Discrete Event Simulation (DES) model to train and test a Deep Reinforcement Learning (DRL) approach for managing the variability of real working conditions caused by dynamic events such as the arrival of new jobs and machine failures. The authors emphasize the use of DES to evaluate artificial intelligence methods for solving the Dynamic Flexible Job Shop Scheduling Problem (DFJSSP). In the DFJSSP, jobs may arrive at various times throughout the planning horizon, rather than all being present in the shop at the start of the scheduling. The authors demonstrate that an intelligent agent, based on DRL and a neural network (NN), can gather data about the current state of the production system and make real-time scheduling decisions. The DES model of the production system facilitates simulating the environment with which the intelligent agent interacts, enabling the testing and training of novel artificial intelligence-based methods. Essentially, a Reinforcement Learning (RL) problem can be described as a Markov decision process in which an agent begins in a specific state within its environment. At each time step, the agent must take an action. Subsequently, based on the resulting state transitions, the agent receives a reward. In the context of scheduling, the system's state is represented as a vector of state variables at each time step, and the agent has a set of possible actions to choose from. If a performance metric improves within the scheduling environment, the agent receives a positive reward. The scheduling environment is simulated by DES, allowing the simulator to be viewed as a function that outputs the performance metric. The reward is determined, forming the tuple (initial state, action, next state, reward). A training algorithm uses this tuple to adjust the NN's parameters. The outcome of the training process is a set of optimized parameters which are updated based on the newly observed conditions.

[36] addressed the Job Shop Scheduling Problem (JSSP). According to the authors, nature-inspired algorithms such as Particle Swarm Optimization (PSO) and the Firefly Algorithm (FA) are among the most effective optimization techniques. In their study, the authors employed the FA to solve the JSSP. The behavior of fireflies was modeled to tackle the problem, with key characteristics including: (a) all fireflies are unisex; (b) each firefly is attracted only to fireflies that are brighter than itself, with the intensity of attraction proportional to the firefly's brightness and decreasing with distance; the brightest firefly moves randomly; and (c) each firefly's brightness corresponds to the quality of its solution, typically proportional to the objective function. For the JSSP, the brightness of each firefly is

determined by the makespan. The Cartesian distance between fireflies is calculated using their spatial coordinates to measure separation. A formula was proposed to compute the attractiveness between fireflies, and the movement of each firefly was accordingly determined. A computational experiment was conducted using twenty-five benchmark JSSP datasets from [37]. The results were compared with those reported in [38] and [39]. Out of the 25 problems, 21 were successfully and precisely matched with the best-known solutions. (actual benchmarks), while three problems achieved a 96% confidence level and were nearly optimal.

[40] presented hybrid and multistage structures to enhance a Genetic Algorithm (GA) for addressing the FJSSP. In the initial phase, the GA determines the order in which operations are assigned. In the second phase, a greedy method assigns machines by selecting the one with the shortest completion time for each operation, based on the machine's current load and processing time. Selection at both stages is conducted through a k-way tournament. As described by the authors, a k-portion of the population, consisting of k individuals, is randomly chosen for the k-way tournament, and the solution vector with the highest fitness is copied to the reproduction pool. This process is repeated until the reproduction pool reaches the population size. Stage one reproduction involves sequencing crossover and operation swap. In phase two, all operators from stage one are applied, along with assignment mutation and assignment crossover. Their results were compared with the best performance reported by 21 researchers. Two sets of benchmark problems were used for comparison, specifically the instances from references [9] and [23]. The conclusions indicate that their approach significantly outperforms existing methods and is a promising technique for solving large-scale FJSSP problems.

[41] focused on solving the multi-objective dynamic flexible job shop scheduling problem (MO-DFJSSP). The authors developed a multi-objective genetic programming-based hyper-heuristic (MO-GPHH) aimed at producing efficient scheduling policies offline for rapid online application. A scheduling policy for the MO-DFJSSP consists of two decision rules in a scheduling policy for the MO-DFJSSP: a job sequencing rule (JSR) and a machine assignment rule (MAR). This research considered three objectives to minimize: mean flow time, maximum tardiness, and mean weighted tardiness. Additionally, a discrete event simulation model for the MO-DFJSSP was used as a testbed for conducting experiments. The authors noted that other evolutionary algorithms represent population members as fixed-length strings of genes, whereas in genetic programming (GP), the population members are represented as trees of varying lengths. During the evaluation stage of the GP algorithm, each population member, after being decoded into decision rules, is applied to the relevant decision points in the simulation model. Throughout the evolution stage outcomes gathered from the simulation model are fed back to the GP algorithm to assign fitness values to the population members. Three popular metrics were employed to evaluate the performance of the proposed methods: hypervolume ratio (HVR) as detailed in [42], Inverted Generational Distance (IGD) proposed by [43], and the Spacing indicator described in [44]. This study used 320 benchmark scheduling policies from [45] and [46] to validate the efficacy of the proposed approach. The results showed that the nondominated scheduling policy solutions evolved by MO-GPHH dominate all previously available scheduling policies across all metrics.

[47] utilized a constraint programming (CP) model to address two distinct variations of the FJSSP: one where the task sequence within a job is mandatory, and another where the tasks within a job can be executed in any order. It is important to clarify that each task consists of a collection of operations. The authors emphasize that, in both problem types, the sequence in which tasks are completed is mandatory. A particular characteristic of both problem variations is that a machine can have multiple workplaces of various kinds, but only one workplace can be active at a time, and only one workplace can produce the product at a time. Therefore, the main objective of this paper was to schedule the tasks for each product (job). The authors applied heuristic techniques to identify feasible solutions. The first heuristic ranks are operations based on the order in which they were assigned to the machine. The second heuristic ranks operations according to the machine's end time in the order they were assigned. For the experimental tests, 50 instances were used, with 1 to 10 jobs, each consisting of 1 to 5 tasks, and each task requiring 1 to 3 operations randomly selected. Additionally, there were 226 workplaces, and processing times ranged from 1 to 60 minutes. A comparison between the heuristics and the CPLEX solver was conducted. Both heuristics demonstrated competitive performance in most instances.

[48] conducted a study within the framework of zero-defect manufacturing, where minimizing defective parts and machinery breakdowns is a critical concern. The authors focused on modifying existing heuristics to develop improved versions for generating initial solutions, which were then optimized using a tabu search algorithm. The proposed heuristics consist of two components: prioritizing orders and distributing operations among available machines. Orders were prioritized using two heuristics: Come, First Served (FCFS) and OrderSequence (OS), which evaluates the importance of orders based on client profile, order volume, and due date. The heuristics used to assign operations to machines included Machine Cost (MC), Shortest Processing Time (SPT), Sum of Shortest Processing Times (SUMPS) and Earliest Completion Machine (ECM). For experimental validation, an actual industrial dataset from the semiconductor industry was employed. The simulations focused on the assembly phase of a custom printed circuit board (PCB). Results indicated that the OS-ECM combination produced the most balanced schedules compared to other heuristic pairings.

[49] presented a novel approach using tabu search (TS) as the local search method within a multi-objective evolutionary algorithm (MOEA) to solve the FJSSP. The authors emphasize two key aspects in their research: first, TS is employed both in the mutation operator and as a local search technique; second, stagnation is avoided by utilizing the hypervolume indicator proposed by [50]. A tuple (u, v) is used as the solution representation based on [51], where v denotes the machine assignment for operations, and u represents the sequence of operations. Specifically, u is an integer vector in which each job's operations are indicated by the corresponding job number. Each integer in the machine assignment vector v specifies the machine assigned to each operation in sequence. The initial population is generated with machine assignment vectors using two assignment rules, while the operation sequence vectors are created by combining three established dispatching rules. Selection is performed via tournament selection. The crossover and mutation operators are described in the manuscript based on [51]. However, for the machine assignment vectors, the mutation operator is a hybrid of the approach in [51], and TS. A non-dominated sorting and the crowding distance operator from [52] are applied during replacement. The authors tested their method on the well-known datasets from [9] and compared their solutions to those produced by the algorithms in [53]. Their method was able to find better solutions and dominate multiple points identified by the two algorithms, as well as discover solutions that were neither contained in nor dominated by the Pareto front approximations found by other algorithms, although it did not produce better overall solutions. Finally, a non-parametric statistical test was conducted to determine whether the hypervolume differences obtained using their strategy were significantly superior to those obtained by one of the other strategies.

[54] proposed a memetic algorithm (ME) for the FJSSP, utilizing two chromosomes as the solution vector.

Both chromosomes are encoded as lists of integers. The first chromosome represents the assignment of each machine operation, with integer values ranging from 0 to $k - 1$, where k is the number of available machines. The second chromosome encodes the sequencing of the assigned operations, using values between 0 and $n! - 1$ to indicate the order of jobs on each machine, where n is the number of jobs to be scheduled. The selection process is hierarchical, following the guidelines specified for the multiobjective scheme. A uniform crossover was employed in this approach, and reciprocal gene exchange was selected as the mutation method. SA was used as a local search technique to complement the genetic stage. To evaluate the efficiency of the proposed method, several examples were drawn from [33]. Their technique demonstrated a balanced distribution of results and rapid convergence to regions near the optimal solution. The authors concluded that this method is effective for solving the FJSSP and plan to apply it to other combinatorial problems in the future.

[55] consideration of a suitable modeling and simulation algorithm is essential to accurately represent a specific plant throughout the optimization process. First, a production system was detailed within the framework of Timed Coloured Petri Nets (TCPN), as described by [56], to model flexible CNC machines capable of simultaneously working with multiple materials and tools. Second, the authors implemented a simulation algorithm using the TCPN model to calculate system throughput. Finally, they applied a PSO scheme integrated with the TCPN simulation to optimize job type sequencing, increase system throughput, and determine the quantity of each job type entering an ophthalmic lens production system. The TCPN effectively managed the input buffer, where jobs are stored in an area

with a limited number of processing instruments while awaiting processing, as well as the working cell where jobs are completed.

Additionally, the TCPN was able to represent the start of a job's activity on the CNC machines, its completion, and its

transfer to the next machine. The authors simulated the TCPN by organizing task categories according to two well-known dispatching rules: Shortest Processing Time (SPT) and Longest Processing Time (LPT), as detailed in [57]. The case study of large-scale ophthalmic lens manufacturing demonstrated that the proposed approach

increased system throughput by over 23% within a single hour.

[58] presented a hybrid metaheuristic algorithm that combines a PSO procedure with elements of the TS metaheuristic to address the FJSSP.

Twelve benchmark test examples from various reference sources were experimentally evaluated to demonstrate the effectiveness of the proposed approach. The test results were compared with those obtained from a genetic algorithm and the results reported in the reference sources. This comparison highlights the strong performance of the proposed method.

[59] described a PSO and LS approach for the multi-objective flexible job-shop scheduling problem (MO-FJSSP). In this research, each particle in the PSO scheme is represented by two components. Part A encodes the routing policy, specifying machine assignments for operations, while Part B represents the sequencing of operations on each machine, indicating operation priorities. The priority level of each operation is determined by the value in Part A, which corresponds to the machine selected for that operation. In the first stage, the authors arranged the machines available for each operation in ascending order of processing time. The priority (random key) influences the scheduling and structure of the solution represented in Part B of the particle. Each element in the particle corresponds to an operation, and its associated value indicates the operation's priority. This strategy was compared with other methods proposed for solving the FJSSP, which were categorized into two groups: Pareto-based approaches and weighted sum of objectives. The results demonstrated that, compared to existing MO-FJSSP methods, the proposed strategy is both competitive and effective.

[60] created a Variable Neighborhood Descent (VND) algorithm to solve FJSSP large instances. To validate the results of the VND algorithm, the group of data instances published in [23] was used.

[61] pointed out that generating solutions for the FJSSP is a challenging problem to apply in actual production settings because of the numerous presumptions that must be met, particularly when the workload is inconsistent or insufficient to sustain the manufacturing process for an extended period, resulting in sporadic idle periods. Therefore, the authors developed an EDA-based approach coupled with a simulation model to find a solution and implement it. The proposed scheme was implemented in a steel door manufacturing facility. The authors used two types of vectors for solution representation. The first type is built to assign the processing sequence of operations to machines, and the second for the assignment of operations to machines. Different probabilistic models were employed to estimate the relations among the positions of operations in the processing sequence of operations on machines and to estimate the interactions among possible assignments of operations on machines. The simulation model provided the necessary feedback on the performance metrics in the facility. The results were compared with those of the GA. The authors concluded that using the proposed approach (called SEDA), different machines allocated to different schedules can significantly increase shop performance.

[62] highlighted several actual manufacturing and service businesses have been forced to employ different equipment or procedures for each operation, and there are alternate process plans available for each job. It is known as flexible job shop scheduling problem with process plan flexibility (FJSSP-PPF). To tackle this problem, the authors employed an EDA coupled with the generalized Mallows distribution (MEDA), where three different vectors were used. The first vector is called the process plan vector; each element displays the associated selected job, and its length is equal to the total number of jobs. The second vector is called the operation sequence vector; each element in it has

a non-repeated integer value, and its length equals the total number of operations, that is, this vector is a representation based on permutations. The machine assignment vector is the third vector, where each element corresponds to the machine chosen for each operation. After building the initial population, a Pareto-approach is considered to select the best candidates to compute the generalized Mallows distribution. New offspring are produced from the previous distribution of the population. This process continues until the number of generations is met. To validate the scientific relevance of this study, the authors compared the MEDA results with others in some general and standard benchmarking datasets such as the [63] instances, from abz5 to abz9 specifically; the [64] instances, ft06, ft10, and ft20; the [65] instances, from la01 to la40; the [66] instances, from orb01 to orb10; the [67] instances, from swv01 to swv20; and finally, the [68] instances, called yn, taking four instances from yn1 to yn4. In addition, the multi-objective algorithms proposed by [69] called NSGA, and by [52] called NSGA-II, are proposed as a benchmark for comparison with the MEDA scheme. Furthermore, a few algorithms were suggested as standards for comparison. with the MEDA scheme. The simulation-based genetic algorithm presented by [70], bat algorithm proposed by [71], genetic algorithm and tabu search detailed by [72], and genetic algorithm designed by [73]. This study concluded that an accurate estimate of the operations is provided by the MEDA at the sequence vector locations for the FJSSP-PPF.

[74] presented a comprehensive literature review of swarm intelligence (SI) and evolutionary algorithms (EA) for solving the FJSSP. The authors indicated that the effectiveness of encoding and decoding, as well as the ease of integrating SI and EA with local search operators, are all affected by the design of high-quality encoding and decoding schemes. The authors highlighted four popular methods for encoding and decoding. In addition, the authors explained two main improvement strategies in SI and EA, namely, Initializing Strategies and local search operators. Based on their literature review, GA was the most popular and appeared in 46 journal papers. The second most popular was PSO, with more than 10 journal papers. The authors highlighted emerging algorithms, such as the Migrating Birds Optimizer (MBO), Imperialist Competitive Algorithm (ICA), Shuffled Frog-Leaping algorithm (SFLA), Social Spider Optimization (SSO), Virus Optimization Algorithm (VOA), CRO, and FA scheme.

This review highlights the shortcomings of the state-of-the-art. Specifically, the EDA scheme has been utilized as a proposed solution for optimization problems such as the FJSSP. In the EDA, a probabilistic model must be built to obtain offspring. The EDA performance is directly related to the quality of the model used to generate the offspring. A more accurate probabilistic model is the goal of developing any EDA. One approach is to compute different estimates of the relationships between variables to create dependency graphs by considering specific statistical metrics. The Univariate Marginal Distribution Algorithm (UMDA) depicted by [75], the Mutual Information Maximizing Input Clustering algorithm (MIMIC) detailed by [76], the Combining Optimizers with Mutual Information Trees algorithm (COMIT) discussed in [77], and the Bayesian Optimization Algorithm (BOA) detailed in [78] are examples in this category.

Another way to establish relationships between variables is through distance-based ranking models. These models fall into a new category of models. The generalized Mallows distribution (GMD) explained by [79] is the most representative distribution. In general, in the EDA for the FJSSP, a solution contains a permutation-based representation of integers for the processing sequence of operations on the machines. Then, using the GMD to solve the FJSSP, each sequence will produce a ranking. If ranking k operations is required, the GMD can be used according to certain criteria.

The second category is the radial distribution models. The radial distribution of hydrogen shown in [80] appears to be the most prevalent. The offspring were created using an atomic orbital generation algorithm. This function is analytically well-defined hydrogen.

There is a need to enhance EDA based on prior classification. Researchers and practitioners are still drawn to creating novel probability models. Consequently, the suggested EDA produces solutions based on the Dirichlet distribution. Table 1 summarizes the current research

Table 1. Current research.

Research	Problem	Approach	Objective	Instances	Comparison with
[8]	FJSSP	ABC	makespan	[9]	[10], [11], [12]
[13]	FJSSP with dynamic issues	ABC	makespan	[9]	[14], [15], [16], [17], [18], [19]
[20]	FJSSP	Hybrid, Ant colony & LS	makespan	[23]	[21], [22]
[24]	DFJSSP	CRO	makespan	[9], [23], [25]	[26], [27]
[28]	FJSSP	CS	makespan	[23]	Standard CS
[29]	FJSSP	Hybrid, CS & simulation	makespan	[9]	Mathematical model
[30]	FJSSP	HDE	makespan	[9], [23], [33]	[16]
[32]	FJSSP	DE	makespan	[9], [33], [34]	
[35]	DFJSSP	Hybrid, DRL & simulation	makespan		
[36]	JSSP	FA	makespan	[37]	[38], [39]
[40]	FJSSP	GA	makespan	[9], [23]	
[41]	MO-DFJSSP	GP	Hypervolume ratio, Inverted generational distance, & Spacing indicator	[45], [46]	
[47]	FJSSP	CP	makespan	Random instances	CPLEX
[48]	FJSSP	Heuristics	Defected parts	Industrial data	
[49]	FJSSP	TS	Hypervolume ratio	[9]	[53]
[54]	FJSSP	ME	makespan	[33]	
[55]	FJSSP	Hybrid, TCPN & PSO	throughput	Industrial data	
[58]	FJSSP	Hybrid, PSO & TS	makespan	Diverse sources	GA
[59]	MO-FJSSP	Hybrid, PSO & LS	Pareto approach & weighting summation		
[60]	FJSSP	VND	makespan	[23]	
[61]	FJSSP	Hybrid, EDA & simulation	throughput	Industrial data	GA
[62]	FJSSP-PPF	EDA	Pareto approach	[63], [64], [65], [66], [67], [68]	[52], [69], [70], [71], [72], [73]

As we can see, the development of new EDAs is expected to make them competitive in problems such as the FJSSP. In addition, there is insufficient literature on EDAs to address problems such as the FJSSP. In addition, there is a gap between continuous EDAs for addressing combinatorial problems. There are studies that use continuous values in the solution representation. In [30], solutions were created by uniform distribution and subsequently decoded. The study by [31] is another example, where the initial population is randomized from a number between [0,1]. The contribution of [61] is also present, where the operation sequence vectors are built using continuous values between [0,1]. The aim of this study was to identify mechanisms that improve the performance of EDAs by using parameters with continuous values in the generation of offspring.

3. DEDA — for the FJSSP

3.1 Solution representation

Any solution to the production process should involve both machine assignment and scheduling decisions. Therefore, the assignment of operations to machines and the processing order of jobs can be used to represent solutions.

The total number of elements in the job sequence vector is equal to the number of jobs. For the machine assignment vector, the number of elements is equal to the total number of operations. For each operation, each element corresponds to the selected machine.

3.2 Generation of the population

The job sequence members are generated randomly to enable a wide range of solutions [81]. Machine assignment members are also built randomly. For each operation, one of the feasible machines is randomly selected.

Pseudocode 3.2.1. Generation of job sequence members

Do
$I \leftarrow$ Create an identity permutation of jobs
Do
$u \leftarrow$ Choose randomly a position from I
$v \leftarrow$ Choose randomly a position from I
$I \leftarrow$ Swap job[u] and job[v] in I
Until five exchanges
$D \leftarrow$ Set the member I in the population
Until all members have been generated
Output: job sequence members D

Pseudocode 3.2.2. Generation of machine assignment members

Input: job sequence members D
$O \leftarrow$ Identify the total of operations per job
$L \leftarrow$ Create a list with the initial and final number of operations per job

Do
j ← Read each job of the corresponding member in D
a ← Identify the initial number of operations for j in L
b ← Identify the final number of operations for j in L
For $c = a$ to b
P ← Identify what machines are feasible for the operation c
m ← Choose randomly a feasible machine from P
M ← Set the selected machine m into the machine member
P ← Clear P
Endfor
Until all jobs from each member in D have been read
Output: machine assignment members M

3.3 Fitness

The makespan is computed by DEDA to offer feedback regarding the search progress for the ideal solution. To obtain the makespan for each member of the population, some preliminary steps are taken. First, we identified the total number of operations per job. Second, a consecutive list is created with the initial and final numbers of operations per job. Third, we read each job from the sequence vector and identify the machine selected from the machine assignment vector for each number of operations obtained from the previous list.

Owing to the consecutive list mentioned above, it is possible to track the machine selected for each operation. Subsequenlty, four cases are analyzed to guarantee that the operations of the same job must be processed in their predefined order. Case 1 occurs when the operation of the job is the first operation to be assigned in the schedule, and such an operation is the first operation to be processed in the corresponding machine. Case 2 occurs when the operation of the job is the first operation assigned in the schedule, and such an operation is not the first operation to be processed in the corresponding machine. Case 3 occurs when the operation of the job is not the first operation to be assigned in the schedule, and such an operation is the first operation to be processed in the corresponding machine. Case 4 occurs when the operation of the job is not the first operation to be assigned in the schedule, and such an operation is not the first operation to be processed in the corresponding machine. With the previous conditions, it is feasible to guarantee that the operations of the same job must be processed in their predefined order and to compute the finish time per job. In addition, with the previous conditions, the DEDA scheme allows interleaving of operations from different jobs subjects to precedence and machine availability constraints.

Pseudocode 3.3. Fitness computing

Input: job sequence members D
Input: machine assigment members M
O ← Identify the total of operations per job
L ← Create a list with the initial and final number of operations per job
Do
t ← Set the current time to zero for each job
w ← Set the current time to zero for each machine

Do
j ← Read each job of the corresponding member in D
a ← Identify the initial number of operations for j in L
b ← Identify the final number of operations for j in L
For $c = a$ to b
m ← Read the selected machine for the operation c in M
f ← Identify the applicable case
t, w ← update t and w depending on f
Endfor
Until all jobs have been read
F ← Identify the longer time between the jobs in t
Until all members have been read
Output: Fitness F

An example is provided to show fitness computing. Table 2 lists the alternative machines for each operation in the case of four jobs.

Table 2. An example of FJSSP.

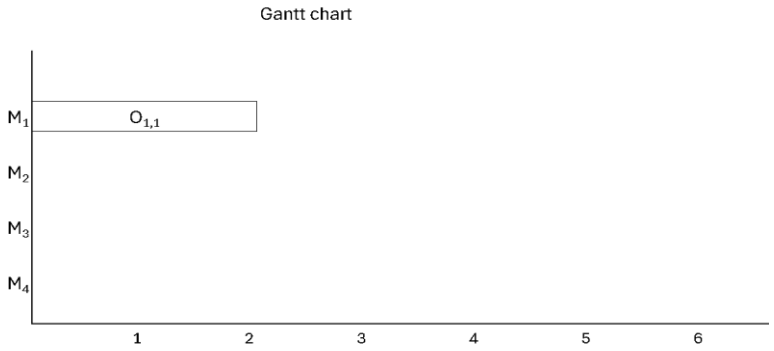
Operations	Processing time machines			
Job,Precedence	M_1	M_2	M_3	M_4
$O_{1,1}$	2	4	3	5
$O_{1,2}$	4	4	-	3
$O_{1,3}$	3	-	-	-
$O_{2,1}$	3	-	3	3
$O_{2,2}$	-	5	6	2
$O_{2,3}$	2	-	-	-
$O_{3,1}$	-	2	3	3
$O_{3,2}$	1	-	-	-
$O_{3,3}$	-	3	3	-
$O_{4,1}$	6	4	5	5
$O_{4,2}$	2	4	3	5
$O_{4,3}$	4	4	-	3

A consecutive list is created with the initial and final numbers of operations per job.

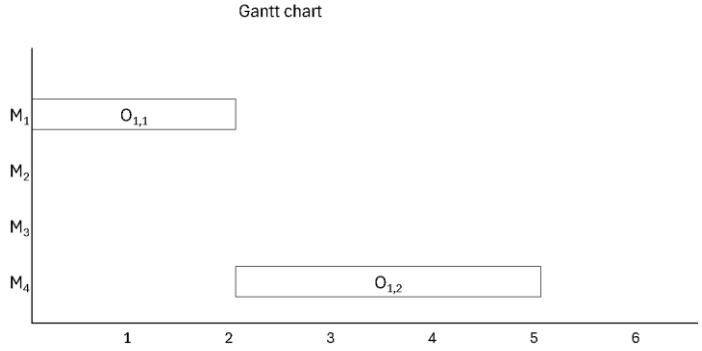
Job,Precedence	Job	# Operation	
$O_{1,1}$	1	1	

$O_{1,2}$	1	2	$1 \prec 2$
$O_{1,3}$	1	3	$2 \prec 3$
$O_{2,1}$	2	4	
$O_{2,2}$	2	5	$4 \prec 5$
$O_{2,3}$	2	6	$5 \prec 6$
$O_{3,1}$	3	7	
$O_{3,2}$	3	8	$7 \prec 8$
$O_{3,3}$	3	9	$8 \prec 9$
$O_{4,1}$	4	10	
$O_{4,2}$	4	11	$10 \prec 11$
$O_{4,3}$	4	12	$11 \prec 12$

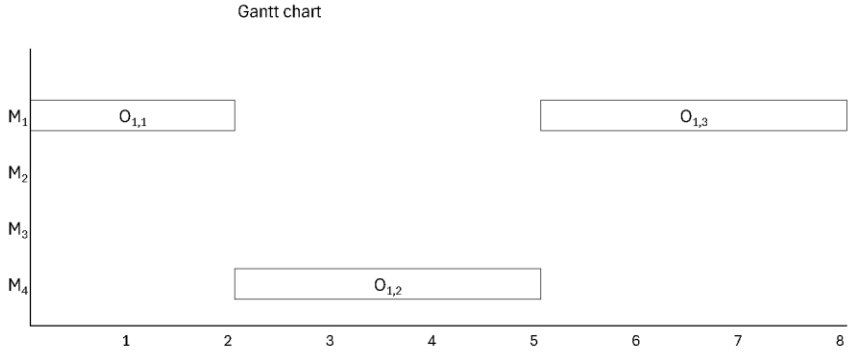
Let the job sequence $J = \{1,2,3,4\}$, and the machine assignment $M = \{1,4,1,3,4,1,3,1,2,2,2,4\}$, that is, machine 1 will be used to process operation 1, machine 4 will be used to process operation 2, and so on. Then, reading the job sequence vector, the first job is 1, and the selected machine for operation 1 is machine 1. The applicable case is Case 1, where the operation of the job is the first operation to be assigned to the schedule, and such an operation is the first operation to be processed in the corresponding machine.



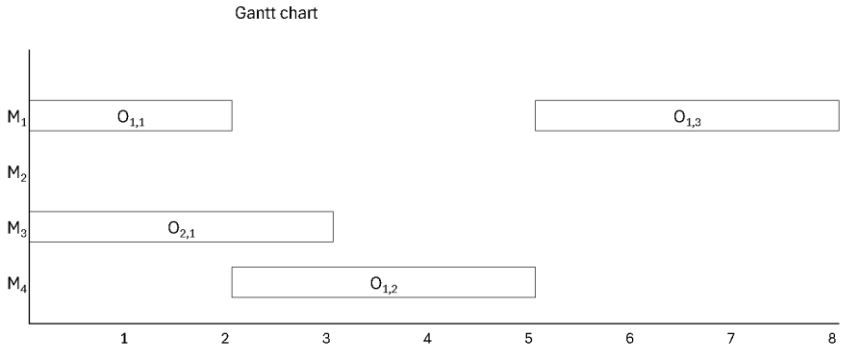
We continue in job 1, and the selected machine for operation 2 is machine 4. Case 3 is applicable because the operation of the job is not the first operation to be assigned in the schedule, and such an operation is the first operation to be processed in the corresponding machine.



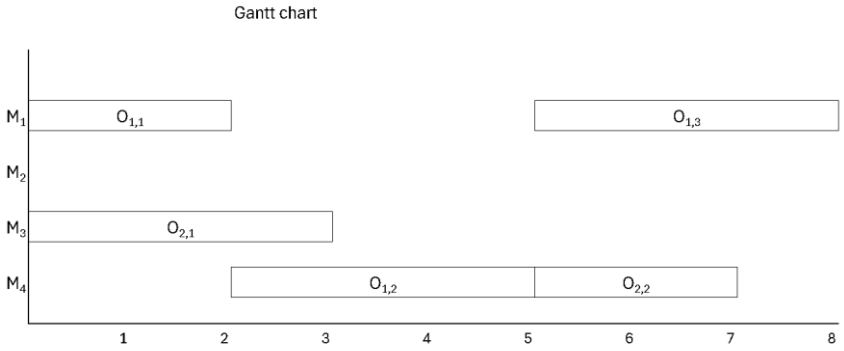
We continue in job 1, and the selected machine for operation 3 is machine 1. The applicable case is Case 4 because the operation of the job is not the first operation to be assigned in the schedule, and such an operation is not the first operation to be processed in the corresponding machine.



Reading the job sequence vector, the next job is job 2, and the selected machine for operation 4 (see the consecutive list above) is machine 3 (see the machine assignment above). Case 1 is applicable.



We continue in job 2, and the selected machine for operation 5 is machine 4. Case 3 is applicable.



This process continues until all operations are incorporated into the procedure.

3.4 Offspring

A new generation is built using a probability model. In this sense, the Dirichlet distribution acts as a mechanism for generating new job-sequence vectors.

3.4.1 The Dirichlet distribution computing

Let a vector contain the probabilities of each possible event outcome $\theta = (\theta_1, \theta_2, \dots \theta_k)$. θ has two crucial characteristics: first, none of the probabilities can be negative, and the total of each entry's probability must equal one. When these requirements are satisfied, the results associated with the events can be described using multinomial distribution. We now determine the probability density of every potential value of θ for a particular process. To do this, we examine every component of θ as a separate variable. The Dirichlet distribution can then be used as the prior distribution of the multinomial distribution.

Based on [82], the probability density is defined by the Dirichlet distribution for a multinomial vector θ. The form of its probability density function is as follows

$$\mathrm{Dir}(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k}\alpha_i)}{\prod_{i=1}^{k}\Gamma(\alpha_i)}\prod_{i=1}^{k}\theta_i^{\alpha_i-1} \quad \text{where } \theta = (\theta_1, \dots \theta_k) \text{ and } \alpha = (\alpha_1, \dots, \alpha_k) \quad (1)$$

vector α, which has the same number of elements as θ, parameterizes the Dirichlet distribution. Therefore, $p(\theta|\alpha)$ can be interpreted as the answer to the question "what is the probability density associated with the multinomial distribution θ, given that our Dirichlet distribution has parameter α?"

For $k = 2$, Dirichlet distribution becomes the beta distribution with parameters α_1 and α_2.

In this study, the starting population of job sequence members is considered an input parameter for calculating the initial parameter vector. This population is viewed as a matrix, with each column indicating the respective position in the job processing sequence. Subsequently, we determine the average for each column and assign these averages to each element of the parameter vector. α.

The rationale for employing the Dirichlet distribution for categorical values stems from Bayesian statistics, where the Dirichlet does not represent the data directly, but rather the occurrence probabilities (proportions) for each category. Even though the data is discrete, the variables that determine the likelihood of that data are continuous, aligning perfectly with Dirichlet's definition. The benefit of employing the Dirichlet distribution is that it enables us to revise beliefs (parameter vector α) in a straightforward and analytical manner by merely summing the observed counts. Dirichlet distribution is recognized for not modeling the categories themselves, but instead for representing the strength of belief regarding the occurrence of those categories. Thus, it is essential to determine which job has the highest likelihood of being assigned to a specific position within a series of jobs utilizing the Dirichlet distribution. The intensity of our belief regarding the occurrence of those categories was of interest. The estimation method that is label-invariant and more closely associated with categorical frequencies will certainly be regarded as future research.

With the obtained estimate of α, we move forward to create m vectors, each consisting of k independent samples. For every vector, these samples are created utilizing gamma distribution. Employing gamma distribution is a highly effective and widely used approach for producing independent samples from a Dirichlet distribution. This approach relies on the connection between gamma random variables and the properties of the Dirichlet distribution [83]. This approach is favored over other rejection techniques as it is more straightforward and quicker, particularly when numerous samples must be produced [84].

To obtain the offspring, assign job 1 to the location with the lowest value in each vector derived from the preceding step. Job 2 is allocated to the second least sized position in each vector, and so forth. The procedure persists until every task has been completed in all the new vectors. Ties are resolved randomly. The procedure of generating offspring allows us to establish a viable job order. Allocating roles to their respective positions allows us to determine the highest likelihood of being assigned to specific roles within a series of jobs through the Dirichlet distribution.

Pseudocode 3.4. Offspring

Input: job sequence members D
$n \leftarrow$ Identify the number of jobs

$s \leftarrow$ Identify the number of members
$\alpha \leftarrow 0$
For $j = 1$ to n
$q \leftarrow 0$
For $i = 1$ to s
Update $q = q + D[i][j]$
Endfor
Update $\alpha[j] = q/s$
Endfor
Do
$sum \leftarrow 0$
$y \leftarrow 0$
For $j = 1$ to n
Read each value $\alpha[j]$ from α
$y[j] \leftarrow$ Generate an independent sample using $Gamma(\alpha[j], 1)$
Update $sum = sum + y[j]$
Endfor
$x \leftarrow 0$
For $j = 1$ to n
Update $x[j] = y[j]/sum$
Endfor
Read each member from x
For $j = 1$ to n
$u \leftarrow$ Identify the position of the smallest element in x
$S^1 \leftarrow$ Assing j to position u in the offspring
$x \leftarrow$ Set a value too large in position u in x
Endfor
Until all members have been generated

Do
Read each member from S^1
$S^2 \leftarrow$ Apply code 3.2.2. for building machine asgmt. offspring
Until all members have been generated
Output: Offspring members S^1, S^2

3.5 Replacement

The educational aspect of the DEDA scheme focuses on the job sequencing element. The machine assignment element is not involved in the learning framework. The DEDA design does not influence the fairness of comparisons with methods that optimize both sequencing and machine allocation, as the key objective function is minimizing the total time taken to finish all tasks (makespan). Enhancing machine allocation primarily targets minimizing alternative objective functions like idle time.

The substitution is conducted through tournament selection. Initially, the entire original population and the offspring are viewed as competitors among themselves. Next, two individuals are selected randomly, and we pick the better one based on the relevant fitness. Third, the selected member is placed into the subsequent generation.

Pseudocode 3.5. Replacement

Input: Parents population D
Input: Offspring members S
Input: Fitness F
Do
$u \leftarrow$ Choose randomly a member from D
$v \leftarrow$ Choose randomly a position from S
If Fitness[u] $<$ Fitness[v] then
$D \leftarrow$ Preserve the member from D and its fitness
Else
$D \leftarrow$ Preserve the member from S and its fitness
Endif
Until all available positions have been filled
Output: new generation D

The main DEDA framework is depicted below

Overall Pseudocode. DEDA framework

$D_0 \leftarrow$ Generate M sequencing vectors and machine assignment vectors
$F_0 \leftarrow$ Compute the fitness for each member from D_0
Best $\leftarrow$ Store the best from D_0

t ≔ 1
Do
$\boldsymbol{\alpha}_t$ ← Compute the parameter $\boldsymbol{\alpha}$ from D_{t-1}
S_t^1 ← Sample offspring sequence vectors from $\boldsymbol{\alpha}_t$
S_t^2 ← Sample offspring machine assignment vectors from S_t^1
S_t ← Set the offspring population $S_t^1 \cup S_t^2$
F_t ← Compute the fitness for each member from S_t
Best ← if apply, update the best from F_t
D_t ← Replace the population from S_t & D_{t-1}
t ≔ t + 1
Until (stopping criterion is met)
Output: Best

4. Results and comparison

The comparison uses the typical benchmarking datasets as input data. The datasets used in this research are the fifteen [9] instances: MK01–MK15. The eighteen [34] instances: 01a–18a. The ten [85] instances: MFJS01–MFJS10. The four [33] instances: K1–K4. All the instances contain the information below

<number of jobs> <number of machines> is the first line.

Next, one line for each job: <number of operations>, followed by <number of machines for this operation> and a pair <machine> <processing time> for each machine.

The machine index begins at 0.

To verify the scientific significance of this work, a few algorithms are suggested as standards to compare with the DEDA scheme. The algorithms are as follows:

The UMDA depicted by [75],

The MIMIC detailed by [76],

The COMIT discussed in [77],

The BOA presented in [78],

The MEDA shown in [62], and

The HEDAMMF explained in [86].

To compare each algorithm's performance, the Relative Percentage Increase (RPI) is calculated. The details for implementing each algorithm in the comparison were taken from the literature. Specifically, the probabilistic models used for each algorithm were coded based on the available theory. However, the population size, the number of generations, the stopping criterion, the number of independent runs per instance, and the seeds were the same for the comparison process.

The population size was set at 1000 members per generation for all the algorithms. The number of generations was set at 100 for all the algorithms. The stopping criterion was set as the number of generations. The number of independent runs per instance was set at 30 for all the algorithms. The seeds were set randomly for all the algorithms. The reason for using 1000 members, as the population size, is that it ensures diversity in evolutionary progress. One

hundred generations are enough to guarantee an efficient search during evolutionary progress, and 30 independent runs are required to execute statistical tests.

Figure 1 shows the distribution of the experimental outcomes for the [9] cases. Many of the DEDA results are centered between 0.10 and 0.60, and most of the results from the other algorithms are situated somewhat apart from the DEDA results. This indicates that compared to the other algorithms used in the comparison, the DEDA is typically closer to the best reported results.

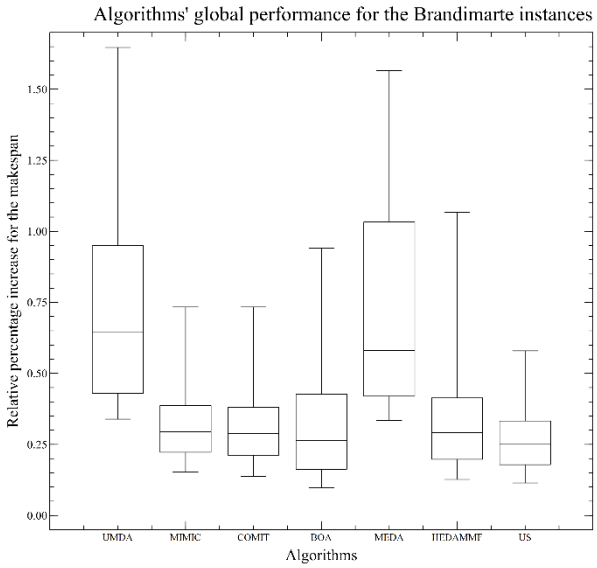


Figure 1. Performance of the DEDA for the [9] instances.

Figure 2 displays the experimental results' distribution for the [34] cases. The results of the other algorithms are positioned very close to 0.10 and 0.40, while the majority of the DEDA results are clustered between 0.05 and 0.25.

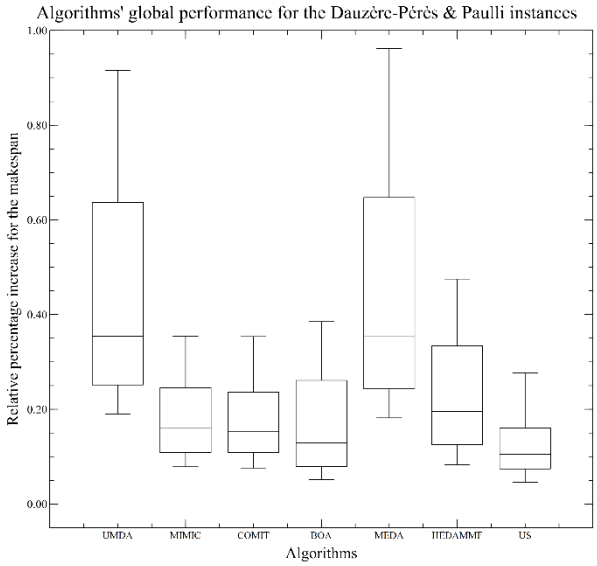


Figure 2. Performance of the DEDA for the [34] instances.

The distribution of the experimental results for the [85] instances is presented in Figure 3. The figure makes it evident that the DEDA approach and the BOA scheme outperform the other methods.

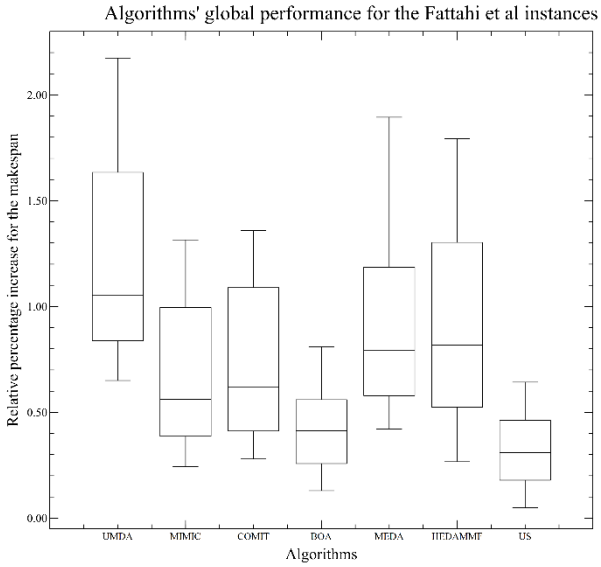


Figure 3. Performance of the DEDA for the [85] instances.

Figure 4 shows the distribution of the experimental outcomes for the [33] cases. Every algorithm used in the comparison performs essentially the same.

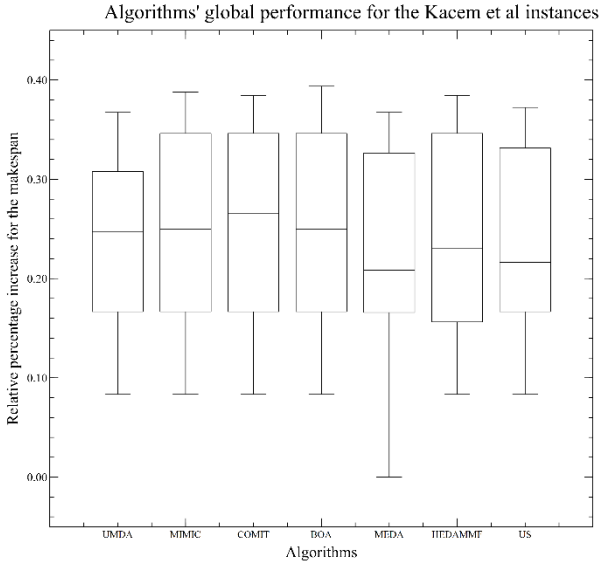


Figure 4. Performance of the DEDA for the [33] instances.

Figure 5 contains the overall performance of all the algorithms. As we can see the DEDA outperforms the comparison.

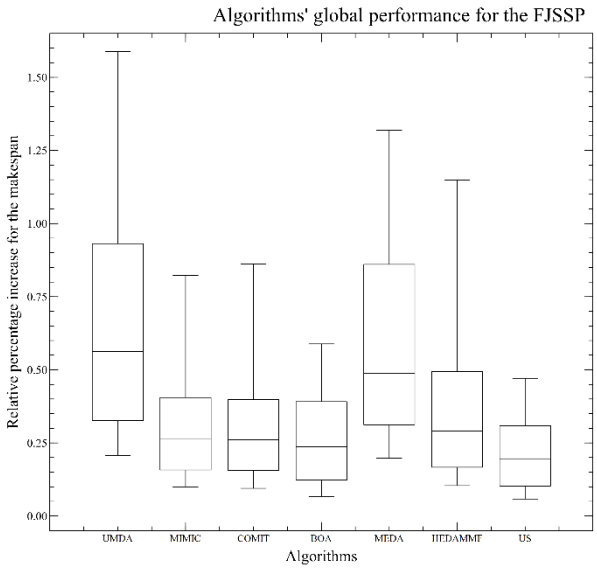


Figure 5. Global performance.

To illustrate the DEDA performance, a full Dunnett statistical test is shown. A Dunnett test for the [9] cases is shown in Figure 6. A statistically significant distinction exists between the DEDA and the BOA, the DEDA and the HEDAMMF, and the DEDA and the UMDA.

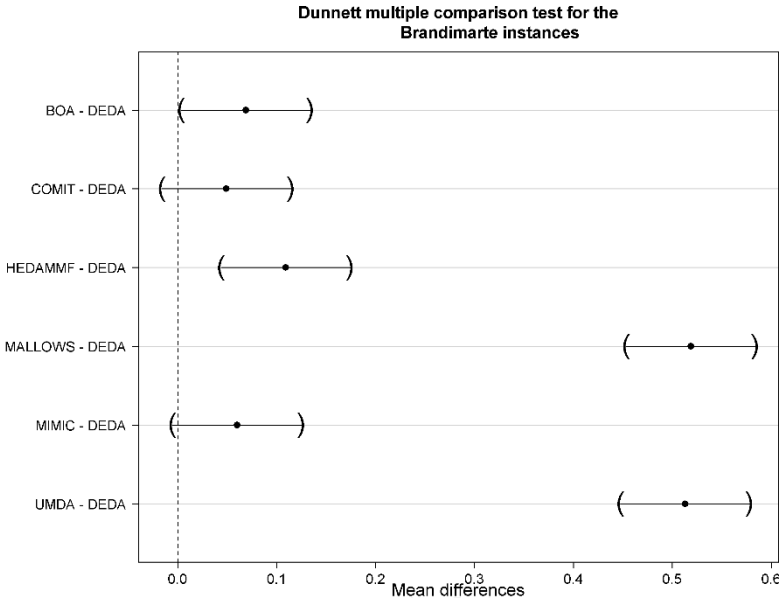


Figure 6. Dunnett test for [9] instances.

A Dunnett test for the [34] cases is shown in Figure 7. The DEDA and the other algorithms differ statistically significantly.

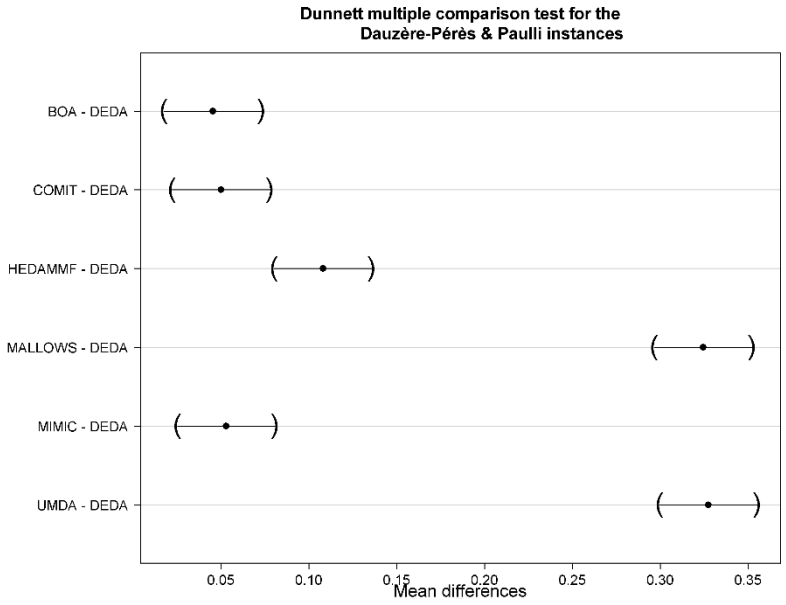


Figure 7. Dunnett test for the [34] instances.

Figure 8 presents a Dunnett test, for the [85] instances. Again, the DEDA and the other algorithms differ statistically significantly.

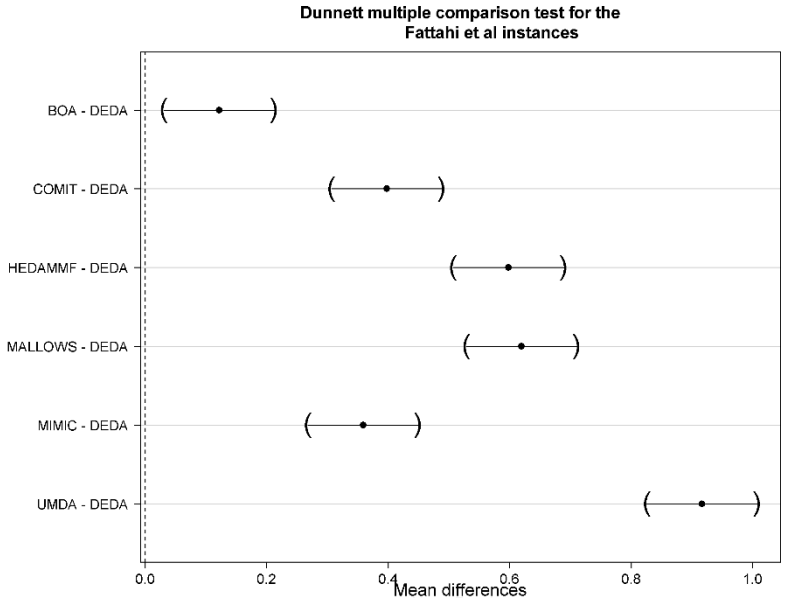


Figure 8. Dunnett test for the [85] instances.

Figure 9 shows a Dunnett test, for the [33] instances. There is no statistically significant distinction between the other algorithms and the DEDA.

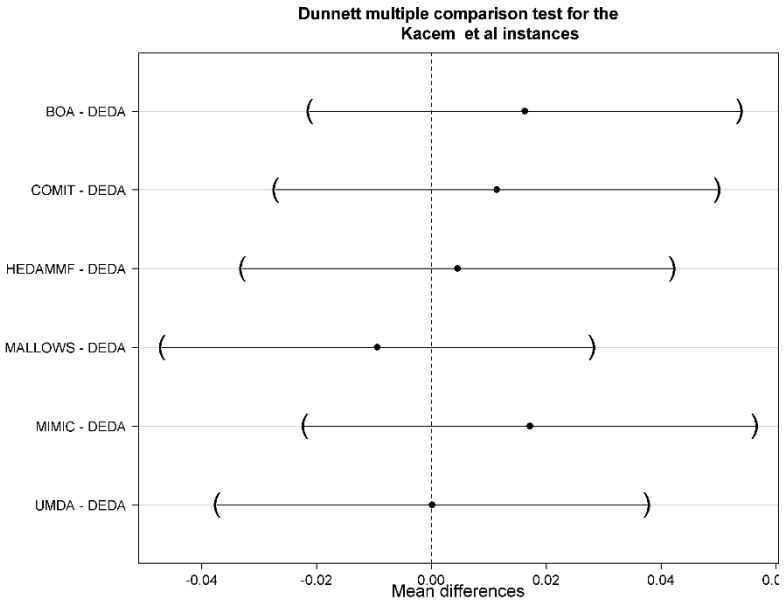


Figure 9. Dunnett test for the [33] instances.

Finally, Figure 10 contains the overall Dunnett test. A statistically significant distinction exists between the DEDA and the other remaining algorithms.

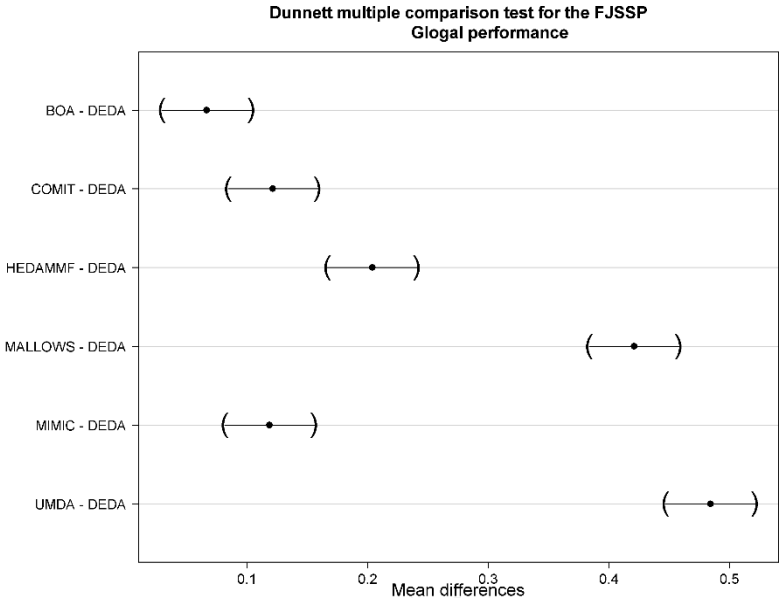


Figure 10. Overall Dunnett test.

5. Conclusions

The FJSSP can be addressed with the DEDA scheme mentioned above. It is a well-known NP-hard issue. The DEDA scheme is competitive, according to the findings presented in Section 4. Since the solution representation for other scheduling problems may be the same, through permutations, a potential future research direction for the DEDA is to apply it to other types of scheduling problems. Other scheduling issues can be considered such as flow shops, open shops, etc. Other combinatorial problems, including vehicle routing scenarios, can also be solved with the DEDA

as a potential future research direction. Thus, to verify its effectiveness, the DEDA scheme should tackle other optimization issues.

Benchmarking refers to the collection of examples utilized in the comparisons. As a result, the application of the Dunnett test is persuasive and well-founded.

The literature should take the DEDA scheme's performance into account.

The DEDA's suggestion to improve offspring performance by utilizing the Dirichlet distribution is significant.

Other greedy methods should be used in subsequent studies to assist the DEDA in identifying better sequencing solutions. To improve the DEDA's performance, more distributions should be considered to produce new offspring.

Lastly, this approach should be used to create application tools for users in real-world scenarios and in practice.

Acknowledgments

We would like to thank each reviewer for their insightful criticism that helped us improve the manuscript.

Author contributions

Conceptualization, R.P.-R.; methodology, R.P.-R.; formal analysis, R.P.-R.; investigation, R.P.-R.; writing—original draft preparation, all authors; writing—review and editing, all authors; visualization, all authors. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Data Availability Statement:

The data presented in this study are available on request from the corresponding author.

Conflict of interest

The authors declare no conflicts of interest.

REFERENCES

1. D. Blanchard, Supply chain management best practices. John Wiley & Sons, 2010.
2. F. Misni, & L.S. Lee, A review on strategic, tactical and operational decision planning in reverse logistics of green supply chain network design. Journal of Computer and Communications, 5 (2017), 83–104.
3. R.B.M. De Koster, T., Le-Duc, & K.J. Roodbergen, Design and control of warehouse order picking: A literature review. European Journal of Operational Research, 182 (2007), 481–501.
4. K. Il-Choe, & G.P. Sharp, Small parts order picking: design and operation. Technical report, Atlanta, EEUU School of Industrial and Systems Engineering. Georgia Institute of Technology, 1991.
5. S. Boonstoppel, Solving the flexible job shop problem with alternative process plans. Master thesis. Research Institute of Information and Computing Sciences. Utrecht University, 2025.
6. L., Gao, C., Peng, C., Zhou, & P. Li, Solving flexible job shop scheduling problem using general particle swarm optimization. In Proceedings of the 36th CIE Conference on Computers and Industrial Engineering, (2006), 3018–3027.
7. N.J. Escamilla-Serna, J.C. Seck-Tuoh-Mora, J. Medina-Marín, I. Barragán-Vite, & J.R. Corona-Armenta, Pädi Boletín científico de ciencias básicas e ingenierías del ICBI, 10-2 (2022), 56–64. DOI: 10.29057/icbi.v10iEspecial2.8651
8. A. Thammano, & A. Phu-ang, A hybrid artificial bee colony algorithm with local search for flexible job-shop scheduling problem. Procedia Computer Science, 20 (2013), 96–101.
9. P. Brandimarte, Routing and scheduling in a flexible job shop by tabu search. Annals of Operations Research, 41-3 (1993), 157–183.
10. K.Z. Gao, P.N. Suganthan, & T.J. Chua, Pareto-based discrete harmony search algorithm for flexible job shop scheduling, Proceedings of 12th International Conference on Intelligent Systems Design and Applications, (2012), 953–956.
11. S., Habib, A., Rahmati, & M. Zandieh, A new biogeography-based optimization (BBO) algorithm for the flexible job shop scheduling problem. The International Journal of Advanced Manufacturing Technology, 58 (2012), 1115–1129.
12. N.M. Najid, S. Dauzere-Peres, & A. Zaidat, A modified simulated annealing method for flexible job shop scheduling problem, Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 5 (2002), DOI:

10.1109/ICSMC.2002.1176334.
13. I.C. Ferreira, B. Firme, M.S.E. Martins, T. Coito, J. Viegas, J. Figueiredo, & J.M.C. Sousa, Artificial bee colony algorithm applied to dynamic flexible job shop problems. In M.-J. Lesot et al. (Eds.): IPMU 2020, CCIS 1237, (2020), 241–254.
14. M. Cunha, Scheduling of flexible job shop problem in dynamic environments. Master's thesis, Instituto Superior Técnico, 2017.
15. N.B. Ho, J.C. Tay, & E.M.K. Lai, An effective architecture for learning and evolving flexible job-shop schedules. Eur. J. Oper. Res., 179-2 (2007), 316–333.
16. J.Q. Li, Q.K. Pan, P. Suganthan, & T. Chua, A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem. Int. J. Adv. Manuf. Technol., 52(5–8) (2011), 683–697. DOI: 10.1007/s00170-010-2743-y
17. L. Wang, G. Zhou, Y. Xu, S. Wang, & M. Liu, An effective artificial bee colony algorithm for the flexible job-shop scheduling problem. Int. J. Adv. Manuf. Technol., 60(1–4) (2012), 303–315.
18. L.N. Xing, Y.W. Chen, P. Wang, Q.S. Zhao, & J. Xiong, A knowledge-based ant colony optimization for flexible job shop scheduling problems. Appl. Soft Comput., 10-3 (2010), 888–896.
19. M. Yazdani, M. Amiri, & M. Zandieh, Flexible job-shop scheduling with parallel variable neighborhood search algorithm. Expert Syst. Appl., 37-1 (2010), 678–687.
20. W. Torres-Tapia, J.R. Montoya-Torres, J. Ruiz-Meza, & S. Belmokhtar-Berraf, A matheuristic based on ant colony system for the combined flexible jobshop scheduling and vehicle routing problem. IFAC Papers online, 55-10 (2022), 1613–1618.
21. A.A. García-León, & W.F. Torres-Tapia, A hybrid algorithm to minimize regular criteria in the job-shop scheduling problem with maintenance activities, sequence dependent and set-up times. In Communications in Computer and Information Science, (2020), 208–221, Springer, Cham.
22. A.A. García-León, S. Dauzère-Pérès, & Y. Mati, An efficient Pareto approach for solving the multi-objective flexible job-shop scheduling problem with regular criteria. Computers and Operations Research, 108 (2019), 187–200.
23. J. Hurink, B. Jurisch, & M. Thole, Tabu search for the job-shop scheduling problem with multi-purpose machines. OR Spektrum, 15-4 (1994), 205–215.
24. B. Marzouki, O. Belkahla-Driss, & K. Ghédira, Solving distributed and flexible jobshop scheduling problem using a chemical reaction optimization metaheuristic. Procedia Computer Science, 126 (2018), 1424–1433. DOI: 10.1016/j.procs.2018.08.114
25. F.T.S. Chan, S.H. Chung, & P.L.Y. Chan, Application of genetic algorithms with dominant genes in a distributed scheduling problem in flexible manufacturing systems. International Journal of Production Research, 44-3, (2006), 523–543.
26. L. Giovanni, & F.P. Pezzella, An improved genetic algorithm for the distributed and flexible job-shop scheduling problem. European Journal of Operational Research, 200 (2010), 395–408.
27. P.H. Lu, M.C. Wu, H. Tan, Y.H. Peng, & C.F. Chen, A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems. Journal of Intelligent Manufacturing, 29-1 (2018), 19–34, DOI: 10.1007/s10845-015-1083-z.
28. A.T.S. Al-Obaidi, & S.A. Hussein, Two improved cuckoo search algorithms for solving the flexible job-shop scheduling problem. International Journal on Perceptive and Cognitive Computing, 2-2 (2016), 25–31. DOI: 10.31436/ijpcc.v2i2.34
29. R.K. Phanden, L.K. Saharan, & J.A. Erkoyunku, Simulation based cuckoo search optimization algorithm for flexible job shop scheduling problem. In ICIST '18: Proceedings of the International Conference on Intelligent Science and Technology. London, ACM, (2018), 50–5. DOI: 10.1145/3233740.3233752
30. Y. Yuan, & H. Xu, Flexible job shop scheduling using hybrid differential evolution algorithms. Computers & Industrial Engineering, 65 (2013), 246–260.
31. M. Pinedo, Scheduling: Theory, algorithms, and systems. Englewood Cliffs, NJ: Prentice-Hall, 2002.
32. P. Sriboonchandr, N. Kriengkorakot, & P. Kriengkorakot, Improved differential evolution algorithm for flexible job shop scheduling problems. Math. Comput. Appl., 24-3 (2019), 80. DOI: 10.3390/mca24030080
33. I. Kacem, S. Hammadi, & P. Borne, Pareto-optimality approach for flexible, job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. Mathematics and computers in simulation, 60(3-5) (2002), 245–276.
34. S. Dauzère-Pérès, & J. Paulli, Solving the general multiprocessor job-shop scheduling problem. Technical report, Rotterdam School of Management, Erasmus Universiteit Rotterdam, 1994.
35. L. Tiacci, & A. Rossi, A discrete event simulator to implement deep reinforcement learning for the dynamic flexible job shop scheduling problem. Simulation Modelling Practice and Theory, 134 (2024), DOI: 10.1016/j.simpat.2024.102948
36. K.C. Udaiyakumara, & M. Chandrasekaran, Application of firefly algorithm in job shop scheduling problem for minimization of makespan. Procedia Engineering, 97 (2014), 1798 – 1807.
37. J.E. Beasley, OR-library: distributing test problems by electronic mail. Journal of the Operational Research Society, 41 (1990), 1069–1072.
38. M. Chandirasekaran, P. Ashokan, S. Kumanan, S. Umamaheswari, Multi objective optimization of job shop scheduling using sheep flocks heredity model algorithm. Internation Journal of Manufacturing Science and Technology, 9-12 (2007), 47–54.

39. M. Chandirasekaran, P. Ashokan, S. Kumanan, S. Umamaheswari, Application of selective breeding algorithm for solving job shop scheduling problems. Internation Journal of Manufacturing Science and Technology, 9-12 (2007), 103–118.
40. D. Rooyani, & F.M. Defersha, An efficient two-stage genetic algorithm for flexible job-shop scheduling. IFAC PapersOnLine, 52-13 (2019), 2519–2524.
41. Y. Zhou, & J-J. Yang, Automatic design of scheduling policies for dynamic flexible job shop scheduling by multi-objective genetic programming based hyper-heuristic. Procedia CIRP, 79 (2019), 439–444.
42. D.A. Van Veldhuizen, & G.B. Lamont, Multi-objective evolutionary algorithm test suites. ACM Symp. Appl. Comput, 8 (1999), 351–357.
43. C.A. Coello-Coello, & N.C. Cortés, Solving multi-objective optimization problems using an artificial immune system. Genet Program Evol M 2005, 6 (2005), 163–190.
44. J.R. Schott, Fault tolerant design using single and multicriteria genetic algorithm optimization. Master's Thesis, Massachusetts Institute of Technology, 1995.
45. T.C. Chiang, Enhancing rule-based scheduling in wafer fabrication facilities by evolutionary algorithms: Review and opportunity. Comput Ind Eng, 64 (2013), 524–535.
46. K.R. Baker, Sequencing Rules and Due-Date Assignments in a Job Shop. Management Science, 39 (1984), 1093–1104.
47. K. Czerniachowsk, Heuristics for the flexible job shop scheduling problem with a single active workplace on a machine in which different workplaces are available. Procedia Computer Science, 270 (2025), 900–909. DOI: 10.1016/j.procs.2025.09.210
48. F. Psarommatis, M. Vuichard, & D. Kiritsis, Improved heuristics algorithms for re-scheduling flexible jobs shops in the era of zero defect manufacturing. Procedia Manufacturing, 51 (2020), 1485–1490. DOI: 10.1016/j.promfg.2020.10.206
49. M. Kefalas, S. Limmer, A. Apostolidis, M. Olhofer, M. Emmerich, & T. Bäck, A tabu search-based memetic algorithm for the multi-objective flexible job shop scheduling problem. In Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic. ACM, New York, NY, USA, 9 pages. DOI: 10.1145/3319619.3326817
50. M. Emmerich, N. Beume, & B. Naujoks, An EMO algorithm using the hypervolume measure as selection criterion. In Evolutionary Multi-Criterion Optimization, Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J.M., Mattern, F., Mitchell, J.C., Naor, M., Nierstrasz, O., Rangan, C.P., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M.Y., Weikum, G., Coello-Coello, C.A., Hernández-Aguirre, A., & Zitzler E. (Eds.), 3410 (2005), Springer Berlin Heidelberg, Berlin, Heidelberg, 62–76. DOI: 10.1007/978-3-540-31880-4_5
51. X. Wang, L. Gao, C. Zhang, & X. Shao, A multiobjective 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. DOI: 10.1007/s00170-010-2642-2
52. K. Deb, A. Pratap, S. Agarwal, & T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6 (2002), 182–197. DOI: 10.1109/4235.996017
53. Y. Yuan, & H. Xu, Multiobjective flexible job shop scheduling using memetic algorithms. IEEE Transactions on Automation Science and Engineering, 12 (2015), 336–353. DOI: 10.1109/TASE.2013.2274517
54. M. Frutos, A.C. Olivera, & F. Tohmé, A memetic algorithm based on a NSGAII scheme for the flexible job-shop scheduling problem. Ann Oper Res, 181-1 (2010), DOI: 10.1007/s10479-010-0751-9
55. G. Volpe, A.M. Mangini, & M.P. Fanti, Job shop sequencing in manufacturing plants by timed coloured petri nets and particle swarm optimization. IFAC PapersOnLine, 55-28 (2022), 350–355. DOI: 10.1016/j.ifacol.2022.10.365
56. K. Jensen, & L. Kristensen, Coloured Petri Nets. Modelling and validation of concurrent systems. Springer, Berlin, 2009.
57. J.H. Blackstone, D.T. Phillips, & G.L. Hogg, A state-of-the-art survey of dispatching rules for manufacturing job shop operations. International Journal of Production Research, 20-1 (1982), 27–45.
58. A. Toshev, Particle swarm optimization and tabu search hybrid algorithm for flexible job shop scheduling problem – analysis of test results. Cybernetics and Information Technologies, 19-4 (2019), 26–44, DOI: 10.2478/cait-2019-0034
59. G. Moslehi, & M. Mahnam, A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. Int. J. Production Economics, 129 (2011), 14–22. DOI: 10.1016/j.ijpe.2010.08.004
60. A.A. García-León, & W.F. Torres-Tapia, Integrating production and preventive maintenance in the flexible job shop scheduling problem with setup times and sequence-dependent. IFAC PapersOnLine, 59-10 (2025), 751–756. DOI: 10.1016/j.ifacol.2025.09.128
61. R. Pérez-Rodríguez, S. Jöns, A. Hernández-Aguirre, et al., Simulation optimization for a flexible jobshop scheduling problem using an estimation of distribution algorithm. Int J Adv Manuf Technol, 73 (2014), 3–21. DOI: 10.1007/s00170-014-5759-x
62. R. Pérez-Rodríguez, & A. Hernández-Aguirre, A hybrid estimation of distribution algorithm for flexible job-shop scheduling problems with process plan flexibility. Applied Intelligence, 48-10 (2018), 3707–3734. DOI: 10.1007/s10489-018-1160-z
63. J. Adams, E. Balas, & D. Zawack, The shifting bottleneck procedure for job shop scheduling. Manag Sci, 34-3 (1988), 391–401.
64. H. Fisher, & G.L. Thompson, Probabilistic learning combinations of local job-shop scheduling rules. In: Muth, J.F., Thompson, G.L. (eds.). Industrial Scheduling. Prentice Hall, Englewood Cliffs, (1963), 225–251.
65. S. Lawrence, Resource constrained project scheduling: an experimental investigation of heuristic scheduling

techniques(supplement). Graduate School of Industrial Administration. Carnegie-Mellon University, Pittsburgh, 1984.
66. D., Applegate, & W. Cook, A computational study of the job-shop scheduling problem. ORSA J Comput, 3-2 (1991), 149–156.
67. R.H. Storer, S.D. Wu, & Vaccari, R. New search spaces for sequencing problems with application to job shop scheduling. Manag Sci, 38-10 (1992), 1495–1509.
68. T. Yamada, & R. Nakano, A genetic algorithm applicable to large-scale job-shop problems. In: PPSN, 2 (1992), 281–290.
69. N. Srinivas, & K. Deb, Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput, 2-3 (1994), 221–248.
70. R.K. Phanden, & A. Jain, Assessment of makespan performance for flexible process plans in job shop scheduling. IFAC-PapersOnLine, 48-3 (2015), 1948–1953.
71. H. Xu, Z.R. Bao, & T. Zhang, Solving dual flexible job-shop scheduling problem using a bat algorithm. Adv Prod Eng Manag, 12-1 (2017), 5–16.
72. X. Li, & L. Gao, An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. Int J Prod Econ, 174 (2016), 93–110.
73. X. Li, K. Xing, Y. Wu, X. Wang, & J. Luo, Total energy consumption optimization via genetic algorithm in flexible manufacturing systems. Comput Ind Eng, 104 (2017), 188–200.
74. K., Gao, Z., Cao, L., Zhang, Z., Chen, Y., Han, & Q. Pan, A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. IEEE/CAA Journal of Automatica Sinica, 6-4 (2019), 904–916.
75. H. Mühlenbein, & G. Paaß, From recombination of genes to the estimation of distributions I. Binary parameters. In Parallel Problem Solving from Nature—PPSN IV, (1996), 178-187. Springer, Berlin, Heidelberg.
76. J. De Bonet, C. Isbell, & P. Viola, MIMIC: Finding optima by estimation probability densities. In Advances in neural information processing systems, (1997), 424–430.
77. S. Baluja, & S. Davies, Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In D. Fisher (Ed.). Proceedings of the fourteenth international conference on machine learning, (1997), 30–38.
78. M. Pelikan, D.E. Goldberg, & E. Cantú-Paz, BOA: The Bayesian optimization algorithm. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, & R. E. Smith (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99), 1 (1999), 525–532. Orlando, FL: Morgan Kaufmann Publishers.
79. M.A. Fligner, & J.S. Verducci, Distance based ranking models. Journal of the Royal Statistical Society: Series B (Methodological), 48-3 (1986), 359–369.
80. R. Pérez-Rodríguez, A radial hybrid estimation of distribution algorithm for the vehicle routing problem with time windows. Industrial Engineering & Management Systems, 20-2 (2021), 172–183. DOI: 10.7232/iems.2021.20.2.172
81. A. Greenwood, S. Vanguri, B. Eksioglu, P. Jain, T. Hill, J. Miller, & C. Walden, Simulation optimization decision support system for ship panel shop operations. Proceedings of the 2005 Winter Simulation Conference, 2005.
82. S. Sinharay, Continuous Probability Distributions. International Encyclopedia of Education -Third Edition, 2010, 98–102. DOI: 10.1016/B978-0-08-044894-7.01720-6
83. S.V. Stefănescu, Generating Dirichlet random vectors using a rejection property. Kybernetika, 25-6 (1989), 467–475.
84. V.K. Kaishev, & D.S. Dimitrova, Dirichlet bridge sampling for the variance gamma process: pricing path-dependent options. Management Science, 55-3 (2008), 483–496. DOI: 10.1287/mnsc.1080.0953
85. P. Fattahi, M.S. Mehrabad, & F. Jolai, Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. Journal of Intelligent Manufacturing, 18-3 (2007), 331–342.
86. R. Pérez-Rodríguez, An estimation of distribution algorithm for combinatorial optimization problems. International Journal of Industrial Optimization, 3-1 (2022), 47–67. DOI: 10.12928/ijio.v3i1.5862