Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Flexible Job-Shop Scheduling Method Based on an Improved Multi-Objective Artificial Lemming Algorithm

Jin Wu

wujin1026@126.com

Xi'an University of Posts and Telecommunications

Peiyi Sun

Xi'an University of Posts and Telecommunications

Zhenjun GUAN

Xi'an University of Posts and Telecommunications

Research Article

Keywords: flexible job-shop scheduling, multi-objective optimization, artificial lemming algorithm, random-key encoding, Pareto archive

Posted Date: November 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11786069/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License

Additional Declarations: No competing interests reported.

Abstract

A multi-objective flexible job-shop scheduling problem was investigated in which makespan, total energy consumption, and total processing time were minimized simultaneously. To convert continuous search positions into feasible schedules, a two-part random-key representation and an active-insertion decoder were incorporated into an improved artificial lemming algorithm (MSIALA). Multi-start Circle-map initialization, archive-based three-elite guidance, golden-sine search, and segmented elite opposition-based learning were integrated into the search process. MSIALA was evaluated against four comparison algorithms on the Brandimarte Mk01–Mk10 instances, with Mk08 examined in detail. On Mk08, MSIALA achieved a mean hypervolume of 1.039 and a mean inverted generational distance of 0.100. It also obtained the lowest mean Spacing and the highest mean number of Pareto solutions among the compared methods. These results indicate that MSIALA can provide a relatively well-distributed set of alternative schedules on the tested instance.

1. INTRODUCTION

The flexible job-shop scheduling problem (FJSP) extends conventional job-shop scheduling by allowing each operation to be processed on one of several eligible machines[1, 2]. A feasible schedule must determine both machine assignments and operation sequences while satisfying job precedence and machine-capacity constraints. These decisions are interdependent: machine assignments affect processing times and energy consumption, whereas operation sequences determine machine occupancy and job completion times. Makespan alone therefore cannot fully reflect production efficiency and resource use. This study considers the simultaneous minimization of makespan, total energy consumption, and total processing time.

Population initialization, schedule representation, and search mechanisms have been widely investigated for FJSP. Early multi-objective formulations and hybrid methods established a basis for handling competing scheduling objectives[3, 4]. Zhang et al. combined global search, local search, and random generation to construct an initial population and designed problem-specific genetic operators[5]. Li et al. applied an improved grey wolf optimization algorithm to a three-objective FJSP, while Yu et al. incorporated chaotic initialization, adaptive weights, Lévy flights, and genetic operators into a multi-objective sparrow search algorithm[6, 7]. Genetic, neighborhood-search, and particle-swarm hybrids, together with energy-aware formulations, have also been studied[8, 9, 10, 11]. Collectively, these studies indicate that optimization performance depends not only on the search operator but also on the representation and decoding of feasible schedules.

The artificial lemming algorithm updates individuals in a continuous search space[13]. Applying it to multi-objective FJSP raises two related challenges. First, continuous positions must be converted into feasible machine assignments and operation sequences without violating scheduling constraints. Second, the population must explore different objective trade-off regions without becoming concentrated around a single leader or a narrow part of the Pareto front. Addressing these challenges requires the decoding method and the search mechanism to be designed together.

This study therefore proposes a multi-objective improved artificial lemming algorithm (MSIALA). Two-part random-key encoding and active insertion decoding map continuous positions to feasible schedules. Three objective-oriented elites selected from an external Pareto archive jointly guide the population, with cyclic role assignment to reduce persistent bias toward one objective. Different opposition probabilities are applied to the machine-selection and operation-sequencing segments, while multi-start Circle initialization and golden-sine updates support the search at different stages. MSIALA is compared with MOALA, MOAHA, MOMPA, and FISSA on the Brandimarte Mk01–Mk10 instances in terms of the three objectives and solution-set quality; the Mk08 instance is examined in greater detail.

The main contributions of this study are summarized as follows:

(1) A multi-objective improved artificial lemming algorithm (MSIALA) is developed for flexible job-shop scheduling to jointly minimize makespan, total energy consumption, and total processing time. The framework integrates feasible schedule construction with archive-based search to address the coupled decisions of machine assignment and operation sequencing.

(2) An objective-oriented three-elite collaborative guidance strategy is designed to represent different scheduling preferences during the search. Three elites are selected from an external Pareto archive according to the three objectives, and their leadership roles are cyclically reassigned to vary the objective preference of the elite used in the exploitation operator.

(3) A segmented elite opposition-based learning strategy is developed to account for the different roles of machine-selection and operation-sequencing variables. A lower opposition probability is assigned to the machine-selection segment and a higher probability to the operation-sequencing segment, providing a mechanism for exploring alternative sequences while limiting changes to elite machine assignments.

(4) A two-part random-key representation is incorporated with deterministic active insertion decoding to connect continuous position updates with discrete scheduling decisions. The resulting mapping constructs schedules that satisfy eligible-machine assignment, job precedence, and machine-capacity constraints without an additional feasibility-repair procedure.

(5) Multi-start Circle-map initialization and golden-sine search are incorporated as supporting search mechanisms. Independent mapping trajectories are used to initialize the continuous variables, while elite-guided golden-sine updates introduce additional search variation during exploitation.

(6) MSIALA is evaluated against four comparison algorithms on the Brandimarte Mk01–Mk10 benchmark instances. Objective-wise results and hypervolume are reported across the instances, with further analysis of convergence, solution-set distribution, and independent-run variability on Mk08 to characterize the performance of the proposed method.

2. PROBLEM DESCRIPTION

2.1 Flexible job-shop scheduling model

Unlike conventional job-shop scheduling, in which each operation is assigned to a fixed machine, FJSP allows each operation to be processed by one of several eligible machines, potentially with different processing times. It therefore comprises two coupled decisions: machine assignment, which selects an eligible machine for each operation, and operation sequencing, which determines the processing order and start times on the machines. Machine assignments affect feasible sequences; sequences in turn affect machine workloads and job completion times. Together, these decisions determine schedule quality.

The considered system consists of n jobs and m machines. Job $J_j$comprises n_j operations, denoted by $O_{j1},O_{j2},...,O_{j,nj}$. Operation O_{jh} may start only after its predecessor $O_{j,h-1}$ has been completed. Each operation has a nonempty set of eligible machines, $A_{jh}\subseteq\{M1,M2,...,M_m\}$. The processing time of O_{jh} on machine M_k is denoted by p_{jhk}.

The following assumptions are adopted:

1. All jobs are available initially.
2. The operations of each job obey technological precedence.
3. Each operation is assigned to exactly one eligible machine.
4. Each machine processes at most one operation at a time.
5. Processing is nonpreemptive.
6. Transportation and setup times are ignored.
7. Processing times and machine power ratings are known deterministic parameters.

2.2 Mathematical model

Three objectives are considered: the minimization of makespan f_1, total energy consumption f_2, and total processing time f_3. Let v_{jhk} be a binary assignment variable that equals 1 if operation O_{jh} is assigned to machine M_k, and 0 otherwise.

1. Minimization of makespan

Let $C_{j,nj}$ denote the completion time of the final operation of job J_j. The makespan is given by:

$$C_{\max} = \max\{C_1, C_2, ..., C_n\} \quad (1)$$

The objective of minimizing the makespan is defined as:

$$f_1 = \min C_{\max} \quad (2)$$

2. Minimization of total energy consumption

Let W_k denote the processing workload of machine M_k, defined as the sum of the processing times of all operations assigned to it:

$$W_k = \sum_{j=1}^{n} \sum_{h=1}^{n_j} v_{jhk} p_{jhk} \quad (3)$$

Let I_k denote the idle time of machine M_k between the start of its first assigned operation and the completion of its last assigned operation. For a machine with no assigned operations, W_k=0 and I_k=0.

PW_k and PI_k denote the processing power and idle power of machine M_k, respectively. The energy consumption of machine M_k is then given by:

$$E_k = \frac{1}{60}(PW_k W_k + PI_k I_k) \quad (4)$$

The objective of minimizing total energy consumption is defined as:

$$f_2 = \min E = \min \sum_{k=1}^{m} E_k \quad (5)$$

3. Minimization of total processing time

The total processing time TW is the sum of the processing workloads of all machines. Accordingly, the objective of minimizing total processing time is defined as:

$$f_3 = \min TW = \min \sum_{k=1}^{m} W_k \quad (6)$$

4. Constraints

Let S_{jh} and C_{jh} denote the start and completion times of operation O_{jh}respectively. The binary variable v_{jhk} equals 1 if O_{jh} is assigned to machine M_k, and 0 otherwise. A feasible schedule must satisfy the following constraints.

$$\sum_{M_k \in A_{jh}} v_{jhk} = 1, v_{jhk} \in \{0,1\}, M_k \in A_{jh} \quad (7)$$

Eq. (7) requires each operation to be assigned to exactly one eligible machine. Its completion time is determined by its start time and processing time on the selected machine.

$$C_{jh} = S_{jh} + \sum_{M_k \in A_{jh}} p_{jhk} v_{jhk} \quad (8)$$

Operations belonging to the same job must satisfy technological precedence.

$$S_{j,h+1} \geq C_{jh}, h = 1,2,\ldots,n_j - 1 \quad (9)$$

Two distinct operations assigned to the same machine cannot overlap.

$$[S_{jh}, C_{jh}) \cap [S_{uv}, C_{uv}) = \phi, if (j,h) \neq (u,v), v_{jhk} = v_{uvk} = 1 \quad (10)$$

All start times must also be nonnegative.

$$S_{jh} \geq 0 \quad (11)$$

These constraints enforce eligible-machine assignment, processing duration, job precedence, and machine capacity. The random-key decoding and active insertion procedure constructs schedules that satisfy them.

3. DESIGN OF THE ALGORITHM

3.1 Algorithm Improvement Strategy

MSIALA combines a continuous search process with a deterministic scheduling decoder to solve the three-objective FJSP. Each individual consists of two random-key segments representing machine selection and operation-sequencing priorities. The initial population is generated using multi-start Circle mapping, and each individual is decoded into a feasible schedule through active insertion. An external Pareto archive stores nondominated solutions and provides three objective-oriented elites for collaborative guidance. Their leadership roles are cyclically reassigned, and the currently assigned α elite guides the golden-sine exploitation operator when it is activated. Segmented elite opposition-based learning is additionally applied to archive elites, with different transformation probabilities for the machine-selection and operation-sequencing segments. Newly evaluated solutions are considered during archive updating, and crowding distance is used to truncate the archive when its capacity is exceeded. The overall procedure is illustrated in Fig. 1

3.2 Multi-start Circle chaotic initialization

Previous FJSP research has used chaotic initialization to improve initial population diversity and search-space coverage[7]. In conventional Circle-map initialization, however, all variables are generated along a single chaotic trajectory and are therefore recursively correlated. Subsequent discrete decoding may consequently produce duplicate machine-assignment and operation-sequencing schemes. To reduce this trajectory-induced correlation among continuous variables, this study employs multi-start Circle chaotic mapping to initialize the population.

The total number of operations in the FJSP is given by:

$$R = \sum_{j=1}^{n} n_j \quad (12)$$

where n is the number of jobs and n_j is the number of operations in job j. Because the chromosome comprises a machine-selection part and an operation-sequencing part, each of length R, the dimension of each individual is D=2R. For each lemming i in the population, an independent initial value is generated for every dimension:

$$z_{i,g}^{0} \sim U(0,1), i = 1,\ldots,N, g = 1,\ldots,D \quad (13)$$

Each initial value is then iterated independently using the Circle map:

$$z_{i,g}^{t+1} = \mathrm{mod}\left(z_{i,g}^{t} + b - \frac{a}{2\pi}\sin(2\pi z_{i,g}^{t}), 1\right) \quad (14)$$

where $z_{i,g}^{t}$ denotes the chaotic value of dimension g for lemming i after the t-th mapping. a and b are control parameters. In this study, a=0.5 and b=0.2, and the Circle map is iterated 20 times. Finally, the resulting chaotic values are mapped to the search interval of the artificial lemming algorithm:

$$x_{i,g} = l_g + z_{i,g}^{20}(u_g - l_g) \quad (15)$$

When the search interval is [-1,1], the above equation simplifies to:

$$x_{i,g} = 2z_{i,g}^{20} - 1 \quad (16)$$

This yields the initial lemming population. The first *R* dimensions represent the continuous keys for machine selection, while the remaining *R* dimensions represent the priority keys for operation sequencing. These keys are converted into feasible schedules through deterministic decoding. The scatter plot and histogram are shown in Figs. 2(a) and 2(b) respectively.

The population scatter plot shows that the initialized individuals cover the two dimensional search region relatively well, without pronounced large scale clustering or gaps. Moreover, the frequency in each continuous-variable interval is close to the theoretical uniform probability of 0.05, indicating good spatial coverage and approximately uniform marginal distributions in the initial population.

3.3 Encoding and Decoding

1. Machine-Selection Random-Key Encoding and Decoding

Random-key encoding provides a continuous representation for discrete sequencing decisions[14]. To map continuous search positions to machine assignment and operation sequencing in FJSP, a two-part random-key scheme is adopted. Let *R* denote the total number of operations. Each candidate solution therefore has *D*=2*R* dimensions: the first *R* encode machine selection, and the remaining *R* encode operation-sequencing priorities. For lemming *i*, the position vector is expressed as follows:

$$X_i = \left[X_i^{MS}, X_i^{OS}\right] \quad (17)$$

The first *R* dimensions are machine selection keys, and the remaining *R* dimensions are operation-sequencing priority keys. All continuous keys lie in [-1,1].

For operation O_{jh}, suppose the eligible machines are:

$$A_{jh} = \{M_{jh}^1, M_{jh}^2, \ldots, M_{jh}^{K_{jh}}\} \quad (18)$$

where K_{jh} is the number of eligible machines for the *h*th operation of job *j*. The interval [-1,1] is divided into K_{jh} equal subintervals, and the corresponding machine selection key determines the eligible machine index:

$$q_{j,h} = \min\{K_{jh}, \lfloor \frac{(x_{jh}^{MS}+1)K_{jh}}{2} \rfloor + 1\} \quad (19)$$

The q_{jh}th machine in the list of eligible machines is then selected:

$$M_{jh} = A_{jh}(q_{jh}) \quad (20)$$

Thus, a smaller continuous key selects an eligible machine with a lower index, whereas a larger key selects one with a higher index.

2. Operation Sequencing Random Key Encoding and Decoding

The operation-sequencing part uses priority-based random-key decoding. The priority keys assigned to the operations are sorted in ascending order:

$$\pi_i = \arg\operatorname{sort}(X_i^{OS}) \quad (21)$$

The corresponding job indices are then read in sorted order to form a job sequence. The first occurrence of a job denotes its first operation, the second occurrence denotes its second operation, and so on. This preserves the precedence of operations within each job without an additional repair step. If two priority keys are equal, their order is determined by their original positions in the chromosome, ensuring that the same continuous individual always yields the same discrete schedule.

3. Active Insertion Decoding

After obtaining the operation sequence and machine assignments for lemming *i*, the operations are scheduled in sequence. The start time of operation O_{jh} is set to the earliest feasible time that satisfies job precedence and machine-capacity constraints:

$$S_{jh} = \min\{\tau \mid \tau \geq C_{j,h-1}, [\tau, \tau + p_{jhm}) \cap \Omega_m = \phi\} \quad (22)$$

Where τ is a candidate start time, $C_{j,h-1}$ is the completion time of the preceding operation of the same job, p_{jhm} is the processing time of the current operation on machine *m*, and Ω_m denotes the set of time intervals occupied by operations already scheduled on machine M_m. If an existing idle interval can accommodate the operation, it is inserted into the earliest feasible interval; otherwise, it is scheduled after the machine's last operation. The completion time of the operation is:

$$C_{jh} = S_{jh} + p_{jhm} \quad (23)$$

The resulting makespan is:

$$C_{\max} = \max_{j,h} C_{jh} \quad (24)$$

The above procedure is illustrated in Fig. 3.

3.4 Construction of the External Pareto Archive

At iteration *t*, the population and external Pareto archive are denoted by P^t and H^t, respectively. The current population is combined with the archive from iteration *t*-1, and the nondominated solutions are retained as follows:

$$H^t = ND(H^{t-1} \cup P^t) \quad (25)$$

Where, ND(·) denotes nondominated filtering. Solutions with identical objective vectors are removed to avoid duplicate archive entries. Nondominated filtering and external archives are established components of multi-objective evolutionary algorithms[15,16].When the archive exceeds its maximum capacity $H_{\max}$, crowding distance is used to estimate the local density around each nondominated solution. For solution *r*, crowding distance is calculated as follows:

$$CD_r = \sum_{s=1}^{Q} \frac{f_{r+1,s} - f_{r-1,s}}{f_s^{\max} - f_s^{\min}} \quad (26)$$

In this equation, *Q* is the number of objectives, and CD_r is the crowding distance of the *r*-th nondominated solution. A larger crowding distance indicates a sparser region around the solution. When the archive exceeds its capacity, solutions with larger crowding distances are preferentially retained to reduce local clustering and preserve the coverage of the Pareto front.

3.5 Multi-objective Three Elite Collaborative Guidance

The conventional artificial lemming algorithm primarily guides the population using a single global best individual. In multi-objective optimization, however, scheduling solutions generally involve trade-offs, making it difficult for a single global optimum to represent all search directions. Single-elite guidance may cause the population to cluster in a limited region of the Pareto front, reducing solution set diversity. To address this issue, this study combines an external Pareto archive with crowding distance to develop a multi objective three-elite collaborative guidance strategy.

In conventional grey wolf optimization, the population is guided by the α,βandδwolves selected according to fitness.[17]Multi-objective grey wolf optimization instead selects leaders from an external archive because a unique global optimum is generally unavailable[18]. Inspired by this multi-leader mechanism, MSIALA selects one objective-oriented elite for each of the three scheduling objectives and cyclically reassigns their leadership roles. Thus, the three elites represent different objective preferences rather than the three highest-ranked solutions under a single fitness measure.

To account for differences in the units and numerical ranges of the optimization objectives, the objective values in the external archive are first normalized:

$$f_{r,m}=\frac{f_{r,m}-f_m^{\min}}{\max(f_m^{\max}-f_m^{\min},\varepsilon)} \quad (27)$$

In this expression, r indexes the nondominated solutions in the archive, m=1,2,...,M indexes the objectives, and ε is a small positive constant that prevents division by zero. For objective m, the objective oriented evaluation function is defined as:

$$G_{r,m}=\overline{f}_{r,m}+\lambda\sum_{k=1,k\neq m}^{M}\overline{f}_{r,k} \quad (28)$$

The first term emphasizes performance on the target objective, while the second evaluates the solution's overall performance on the remaining objectives. The auxiliary-objective weight is set to λ=0.05 to discourage the selection of an elite solution that performs poorly on the other objectives.

For each of the three optimization objectives considered in this study, an objective-oriented elite is selected from the external archive:

$$L_m=\arg\min_{X_r\in H^t} G_{r,m}, m=1,2,3 \quad (29)$$

Archive members not yet assigned an elite role are prioritized during selection. If multiple objectives identify the same nondominated solution, an unselected archive member with a larger crowding distance is chosen instead. This yields three elites with distinct objective preferences and some spatial separation. Their initial ordering is expressed as:

$$X_\alpha=L_1, X_\beta=L_2, X_\delta=L_3 \quad (30)$$

To prevent the objective oriented elites from remaining permanently assigned to the α, β, and δ roles, a cyclic permutation operator ρ is defined as:

$$\rho(L_1,L_2,L_3)=(L_2,L_3,L_1) \quad (31)$$

The assignment of the three elite roles at iteration t is given by:

$$(X_\alpha^t,X_\beta^t,X_\delta^t)=\rho^{t-1}(L_1,L_2,L_3),\rho^3=I \quad (32)$$

In this expression, t=1,2,...,T, and ρ^3=1 indicates that three cyclic permutations restore the elites to their initial ordering, where I is the identity operator. As the iterations proceed, the three objective-oriented elites assume the α,β and δ roles in the repeating order [L_1,L_2,L_3]→[L_2,L_3,L_1], and [L_3,L_1,L_2]. This rotation reduces the bias toward a particular objective that may arise from fixed elite roles.

For lemming i, candidate positions are calculated under the guidance of each of the three elites:

$$X_{i,l}=X_l-\gamma_l\left|\xi_l X_l-X_i^t\right| \quad (33)$$

$$\gamma_l=2a_t r_{1,l}-a_t,\quad \xi_l=2r_{2,l}, l\in\{\alpha,\beta,\delta\} \quad (34)$$

Where, $a_t=2(1-\frac{t}{T})$, $r_{1,l}$ and $r_{2,l}$ are random numbers in [0,1], and T is the maximum number of iterations. As a_t gradually decreases, the algorithm shifts from global exploration in the early iterations to local exploitation in the later iterations.

Finally, the three candidate positions are combined with equal weights to obtain the collaborative guidance center for lemming i:

$$X_{i,c} = \frac{X_{i,\alpha} + X_{i,\beta} + X_{i,\delta}}{3} \quad (35)$$

3.6 Golden Sine Search and Segmented Elite Opposition-Based Learning

1. Golden Sine Exploitation Operator

During the exploitation stage, the conventional artificial lemming algorithm approaches the current best individual through a standard spiral movement. However, a fixed spiral trajectory may cause repeated searches near that individual. To strengthen local exploitation in later iterations, this study combines golden ratio coefficients with a sinusoidal perturbation and uses the cyclically assigned *α* elite to guide lemming position updates. The golden-sine algorithm provides the underlying search principle[19].

For lemming *i*, a sinusoidal perturbation factor is first generated:

$$s_i = \sin(2\pi r_i), r_i \sim U(0,1) \quad (36)$$

When the algorithm enters the exploitation stage and a random number is less than 0.5, the golden-sine update is applied:

$$Y_i^t = | X_\alpha^t s_i | - \eta_i X_i^t s_i | \varphi X_\alpha^t - (1-\varphi) X_i^t | \quad (37)$$

In Eq.(37), X_i^t denotes the position of lemming *i* at iteration *t*, X_α^t denotes the position of the selected *α* elite, and Y_i^t is the resulting candidate position. The directional factor η_i takes a value of +1 or -1 with equal probability, while the golden-ratio coefficient $\varphi \approx 0.618$ controls the contribution of the current and elite positions.

For the boundary-handling equation, l_g and μ_g denote the lower and upper bounds of dimension *g*, respectively. In this study, l_g=−1 and μ_g=1 for all dimensions.

To keep newly generated positions within the bounds of the continuous decision space, any out-of-bounds components are clipped as follows:

$$Y_{i,g}^t = \min(\mu_g, \max(l_g, Y_{i,g}^t)) \quad (38)$$

2. Segmented Elite Opposition Based Learning

To account for the two-part chromosome structure, this study proposes a segmented elite opposition based learning strategy. Different opposition probabilities are applied to the two segments to explore opposite neighborhoods while preserving favorable scheduling structures. This segment-wise variant builds on the opposition-based learning principle[20].

At the end of each iteration, segmented elite opposition based learning is triggered with probability p_o=0.1. An elite individual X_e in a sparsely populated region is first selected from the current external Pareto archive based on crowding distance. Separate opposition masks *b* are then generated for the machine selection (MS) and operation sequencing (OS) segments. The probability that gene *g* is selected for opposition-based transformation is:

$$p_g = \begin{cases} 0.15, g \in MS \\ 0.25, g \in OS \end{cases} \quad (39)$$

A lower opposition probability is applied to the MS segment to preserve favorable machine assignments in the elite solution, whereas a higher probability is applied to the OS segment to explore alternative operation sequences.

With b_g denoting the opposition indicator for dimension *g* (b_g=1 indicates that opposition is applied), the *g*-th component of the segmented opposition candidate Y_o is defined as:

$$Y_g^o = \begin{cases} l_g + \mu_g - X_g^e, b_g = 1 \\ X_g^e \quad , b_g = 0 \end{cases} \tag{40}$$

The opposition candidate is then converted into a feasible schedule through deterministic random-key decoding and active insertion scheduling, and its three objective values are evaluated. It is considered together with the current population and external Pareto archive during non-dominated filtering. If the candidate contributes a new non-dominated solution, it is retained in the archive. Should the archive exceed its capacity, it is truncated based on crowding distance.

3.8 Experimental Setup

1. Experimental Design

To evaluate the effectiveness of MSIALA for multi-objective flexible job-shop scheduling, the conventional artificial lemming algorithm is used as a baseline. MOAHA and MOMPA, which have shown strong performance in related optimization problems, and FISSA, which is designed for FJSP, are also included for comparison. All algorithms use the same continuous random key representation, deterministic decoding, and active insertion scheduling procedure. The objectives are to minimize makespan, energy consumption, and total processing time. The original studies describe the AHA, MPA, and SSA algorithms underlying the comparison methods[21, 22, 23].

Experiments were conducted in MATLAB R2022a on a computer running 64 bit Windows 11 with an AMD Ryzen 9 8945HX processor and 32 GB of RAM. Each algorithm was run independently 10 times. Within each run, the algorithms used the same random seed to control the influence of randomness on the comparison.

2. Benchmark Instances and Algorithm Parameters

The experiments use the ten Brandimarte flexible job-shop scheduling benchmark instances, Mk01–Mk10. As the instances vary in the numbers of jobs, machines, and operations, they allow the algorithm's performance to be assessed across different problem sizes. Table 1 summarizes their basic characteristics. These benchmark instances were introduced by Brandimarte.

Table 1 Sizes of the benchmark instances

Instance	Number of jobs	Number of machines	Total number of operations
Mk01	10	6	55
Mk02	10	6	58
Mk03	15	8	100
Mk04	15	8	90
Mk05	15	4	106
Mk06	10	15	150
Mk07	20	5	100
Mk08	20	10	225
Mk09	20	10	240
Mk10	20	15	240

To evaluate energy consumption, machine-specific processing power values are assigned. One unit of processing time is assumed to correspond to 1 min, and idle power is set at 20% of processing power. Each machine is assumed to be switched off before its first operation begins and after its final operation is completed; unused machines consume no energy. The machine power parameters are listed in Table 2.

Table 2 Processing and idle power ratings of the machines

Machine	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15
Processing power/kW	4.2	5.1	3.8	6.0	4.6	5.5	3.6	6.3	4.9	5.8	4.0	6.1	4.4	5.3	3.9
Idle power/kW	0.84	1.02	0.76	1.20	0.92	1.10	0.72	1.26	0.98	1.16	0.80	1.22	0.88	1.06	0.78

To ensure a fair comparison, MSIALA, MOALA, MOAHA, MOMPA, and FISSA use the same population size, maximum number of iterations, and external Pareto archive capacity. Each algorithm is run independently 10 times, with a population size of N=200, a maximum of T=300 iterations, and an archive capacity of A_{max}=100. The same random seed is used by all algorithms for each run index to reduce differences arising from random initial conditions.

For MSIALA, the golden-ratio coefficient is set to 0.618. In MOMPA, the movement coefficient P and fish aggregating device probability FADs are set to 0.5 and 0.2, respectively. In FISSA, the proportions of producers and scouts are set to 20% and 10%, respectively. The remaining parameters follow the settings in the respective algorithm implementations.

The five algorithms were first compared on the Mk01–Mk10 instances in terms of the three objectives and nondominated-set quality. The Mk08 instance was then examined in greater detail using three-dimensional solution-set plots, pairwise objective projections, HV convergence curves, distributions of independent-run metrics, and a Gantt chart of a selected schedule.

3.9 Computational Experiments and Analysis

1. Performance Metrics

The objective values were normalized before the solution-set indicators were calculated to account for differences in units and numerical ranges. Nondominated solutions obtained from all algorithms and independent runs were pooled and filtered to construct an empirical reference front. HV and IGD were used to assess convergence and coverage, while Spacing was used to describe the consistency of nearest-neighbor distances. NPS was used to report the number of distinct solutions in the final archive and was interpreted together with the other indicators.[24,25,26]

 Hypervolume (HV)

HV measures the volume of objective space dominated by the nondominated solutions in H^T and bounded by a predefined reference point. It is defined as:

$$HV(H^T) = \mathrm{Vol}\{\bigcup_{x \in H^T}[F(x),(1.1,1.1,1.1)]\} \quad (41)$$

The reference point is set to (1.1,1.1,1.1). HV reflects both the convergence and coverage of the non-dominated solution set. A larger HV indicates better overall quality of the Pareto solution set.

Inverted Generational Distance (IGD)

For each point on the empirical reference front H^{ref}, IGD finds the nearest solution in the algorithm's solution set H^T and then averages these distances.

$$IGD(H^T, H^{ref}) = \frac{\sum_{y \in H^{ref}} \min_{x \in H^T} \| F(x) - Y \|_2}{| H^{ref} |} \quad (42)$$

In this equation, Y is a normalized objective vector on the empirical reference front, and $\|\cdot\|_2$ denotes Euclidean distance. IGD reflects both convergence and coverage of the non-dominated solution set.A smaller value indicates better overall quality of the Pareto solution set.

Spacing

Spacing measures the uniformity of the non-dominated solutions in H^T. It is calculated as the standard deviation of the nearest-neighbor distances d_u, where d_u is the minimum Euclidean distance from the u-th non-dominated solution to any other solution in H^T.

$$Spacing = \sqrt{\frac{1}{N_{ps}-1}\sum_{u=1}^{N_{ps}}(d_u - \bar{d})^2} \quad (43)$$

In Eq.(43), $\bar{d}$ is the mean of all nearest-neighbor distances d_u. Spacing measures the uniformity of the non-dominated solutions in objective space; a smaller value indicates more consistent distances between neighboring solutions and a more uniform Pareto solution set.

Number of Pareto Solutions(NPS)

NPS is the number of unique nondominated solutions in the final external Pareto archive H^T.

$$\mathrm{NPS} = |H^T| \quad (44)$$

NPS indicates the number of trade-off schedules produced by an algorithm. A larger NPS gives decision-makers more nondominated solutions to choose from. Because NPS measures only the size of the solution set, its quality must be assessed alongside HV, IGD, and Spacing.

2. Effect of the Number of Circle-Map Iterations

To determine an appropriate number of Circle-map iterations, K is set to 5, 10, 15, 20, and 25. The mean HV and mean IGD are used to assess the quality of the resulting solution sets. A higher mean HV indicates better convergence and broader coverage of the objective space, while a lower mean IGD indicates that the non-dominated solution set is closer to the empirical reference front.

Table 3 Mean HV and IGD values for different numbers of Circle-map iterations.

K	Mean HV	Mean IGD
5	0.9698	0.1115
10	0.9494	0.1231
15	0.9612	0.1186
20	0.9805	0.1024
25	0.9126	0.1484

Among the five tested settings, K=20 yielded the highest mean HV (0.9805) and the lowest mean IGD (0.1024). Both indicators deteriorated when K was increased to 25. Therefore, 20 Circle-map iterations were used in the subsequent experiments.

3. Comparison of the Five Algorithms

To evaluate MSIALA across flexible job-shop scheduling problems of different sizes, it is compared with the conventional multi-objective artificial lemming algorithm (MOALA), the multi-objective artificial hummingbird algorithm (MOAHA), the multi-objective marine predators algorithm (MOMPA), and an improved sparrow search algorithm for green FJSP (FISSA) on the Brandimarte Mk01–Mk10 instances. MOAHA and MOMPA are recently proposed multi-objective metaheuristics, while FISSA is a discrete optimization algorithm designed for green scheduling.

The instances differ in problem size, objective ranges, and empirical reference fronts. Consequently, HV and IGD values should not be compared directly across instances. The five algorithms are therefore evaluated primarily within each instance. Their objective-wise minima and performance metrics for Mk01–Mk10 are reported in Table 4 and 5.

Table 4 Comparison of MSIALA, MOALA, and MOAHA

Table 5 Comparison of MSIALA, MOMPA, and FISSA

In Table 4 and 5, f_1, f_2, and f_3 denote the minimum makespan, energy consumption, and total processing time, respectively, among the nondominated solutions obtained across all independent runs of each algorithm. These minima are calculated separately for each objective and may correspond to different schedules; they do not imply that a single schedule achieves all three values. Thus, f_1, f_2, and f_3 indicate each algorithm's ability to search in the respective objective directions. HV is averaged over the independent runs and is used to assess the convergence and objective-space coverage of the nondominated solution sets.

Instance	MSIALA				MOALA				MOAHA			
	f_1	f_2	f_3	HV	f_1	f_2	f_3	HV	f_1	f_2	f_3	HV
Mk01	42	12.65	153	1.091	42	12.71	153	0.830	42	12.70	153	1.090
Mk02	28	11.50	140	1.140	30	12.27	150	0.446	29	11.54	142	0.900
Mk03	204	75.27	812	1.260	204	81.12	921	0.585	204	80.71	880	0.769
Mk04	66	26.29	324	1.038	69	27.03	325	0.872	69	26.25	324	0.951
Mk05	177	53.94	672	1.147	178	54.14	673	0.970	177	54.07	672	1.081
Mk06	75	27.06	330	1.168	87	40.26	462	0.188	80	33.99	393	0.489
Mk07	145	52.85	649	1.162	154	54.52	680	0.490	148	54.18	668	0.771
Mk08	523	217.80	2484	1.039	523	221.22	2493	0.739	523	219.43	2484	0.867
Mk09	325	188.72	2210	0.909	354	207.33	2347	0.195	344	205.93	2335	0.266
Mk10	253	153.35	1847	1.1054	270	177.80	2071	0.146	271	172.97	2036	0.240

The Mk01–Mk10 experiments provide an overall comparison across different problem sizes. However, objective-wise minima, mean metric values, and average ranks do not directly show how the algorithms explore the three-objective space or what the resulting schedules look like. Mk08 is therefore selected for a more detailed analysis.

Instance	MSIALA				MOMPA				FISSA			
	f_1	f_2	f_3	HV	f_1	f_2	f_3	HV	f_1	f_2	f_3	HV
Mk01	42	12.65	153	1.091	42	12.62	153	1.035	44	13.16	155	0.596
Mk02	28	11.50	140	1.140	31	12.24	149	0.455	31	12.62	152	0.283
Mk03	204	75.27	812	1.260	204	88.06	998	0.369	204	82.60	941	0.544
Mk04	66	26.29	324	1.038	68	26.69	324	0.944	73	27.19	329	0.749
Mk05	177	53.94	672	1.147	178	54.10	672	1.053	179	54.10	673	0.875
Mk06	75	27.06	330	1.168	89	42.23	463	0.135	89	40.12	466	0.122
Mk07	145	52.85	649	1.162	160	56.93	702	0.305	161	56.83	699	0.246
Mk08	523	217.80	2484	1.039	524	220	2486	0.866	523	220.85	2491	0.769
Mk09	325	188.72	2210	0.909	356	207.13	2345	0.199	356	209.03	2403	0.137
Mk10	253	153.35	1847	1.1054	289	180.73	2109	0.094	284	175.36	2073	0.103

Table 6 Objective values obtained by each algorithm on the Mk08 instance.

Algorithm	f_1	f_2	f_3
MSIALA	523	217.8	2484
MOALA	523	221.22	2493
MOAHA	523	219.43	2484
MOMPA	524	220	2486
FISSA	523	220.85	2491

As shown in Table 6, MSIALA, MOALA, MOAHA, and FISSA all achieve the minimum makespan of 523. MSIALA therefore ties for the best result on this objective. For energy consumption, MSIALA obtains the lowest value of 217.80 kWh, which is 1.55%, 0.74%, 1.00%, and 1.38% lower than those of MOALA, MOAHA, MOMPA, and FISSA, respectively. MSIALA and MOAHA also tie for the

lowest total processing time, while the other algorithms perform slightly less well. These results show that MSIALA achieves the lowest energy consumption, ties for the lowest total processing time, and matches the best makespan.

However, a multi-objective algorithm produces a set of nondominated solutions, so objective-wise minima alone cannot fully characterize its performance. HV, IGD, Spacing, and NPS are therefore used to evaluate solution-set quality. The results are presented in Table 7.

Table 7 Overall performance of the compared algorithms.

Algorithm	Mean HV	Mean IGD	Spacing	NPS
MSIALA	1.039	0.100	0.0178	56.3
MOALA	0.740	0.264	0.0433	26.0
MOAHA	0.867	0.192	0.0371	31.4
MOMPA	0.866	0.175	0.0295	24.9
FISSA	0.769	0.250	0.0506	19.7

On Mk08, MSIALA achieved the highest mean HV (1.039) and the lowest mean IGD (0.100) among the five algorithms. It also obtained the smallest mean Spacing (0.0178) and the largest mean NPS (56.3). Taken together, these results indicate closer agreement with the empirical reference front, more regular solution spacing, and a larger set of alternative schedules on this instance.

4. Pareto Front Analysis

To visually compare the nondominated solution sets, the run with the highest final HV is selected from the ten independent runs of each algorithm. The resulting three-dimensional distributions and two-dimensional projections are shown in Fig. 4 and 5, respectively.

Fig. 4 shows that MSIALA obtains more nondominated solutions over a wider region, extending toward lower energy consumption and total processing time. Although some solutions from the other algorithms overlap with those of MSIALA, most lie in regions with relatively higher energy consumption or processing time, indicating weaker overall coverage across the three objectives.

In the makespan–energy consumption projection, MSIALA's nondominated solutions generally lie in the lower-energy region and form a relatively clear lower-left boundary. This indicates that it can obtain schedules with lower energy consumption at different makespan levels. In the makespan–total processing time projection, some solutions from the five algorithms overlap, but MSIALA offers more alternatives in the lower-processing-time region. In the energy consumption–total processing time projection, its solutions are concentrated toward the lower-left region, indicating favorable trade-offs between these two objectives.

Convergence and Stability Analysis

Fig. 6 shows the mean HV curves of the five algorithms on Mk08. The solid lines represent the means across independent runs, and the shaded bands indicate one standard deviation on either side of those means. MSIALA improved rapidly during the early iterations and maintained the highest mean HV throughout the run. Its mean HV continued to increase gradually during later iterations and reached 1.039 at termination. The curve therefore shows rapid early improvement and sustained later progress.

By comparison, MOALA's mean HV remains below that of MSIALA throughout the run. MOAHA and MOMPA continue to improve in the middle and later iterations but finish with lower HV values, while FISSA improves more gradually. Overall, MSIALA improves solution-set quality quickly at the start and sustains progress in later iterations, showing a favorable balance between global exploration and local exploitation.

Scheduling Solution Analysis

Given the trade-offs among the objectives, the nondominated solution closest to the ideal point is selected from MSIALA's run with the highest HV as the representative compromise schedule. Its Gantt chart is shown in Fig. 8 .

The selected schedule has a makespan of 533, energy consumption of 220.14 kWh, and total processing time of 2514. None of the three objectives is favored at the expense of the others, suggesting that selection by distance to the ideal point identifies a schedule with balanced performance. The Gantt chart also shows no overlapping operations on the same machine, and the operations of each job follow their prescribed precedence order, confirming the schedule's feasibility.

Machine M6 is not assigned any operations in this schedule. This does not affect feasibility: FJSP requires each operation to be assigned to one of its eligible machines, but does not require every machine to be used. Under the specified processing times and machine power parameters, the relevant operations are assigned to other eligible machines. M6's absence is therefore a valid outcome of the machine-assignment decision.

4. Conclusions

An improved multi-objective artificial lemming algorithm was developed for FJSP with makespan, total energy consumption, and total processing time as objectives. A two-part random-key representation and active-insertion decoder were used to map continuous positions to feasible schedules, while archive-based three-elite guidance and segmented opposition-based learning were incorporated into the search. The method was compared with four alternatives on the Brandimarte Mk01–Mk10 instances. On Mk08, it achieved a mean HV of 1.039 and a mean IGD of 0.100, together with the lowest mean Spacing and highest mean NPS among the compared methods. These findings support its effectiveness on the tested instances. Future work will extend the current model to dynamic scheduling environments involving machine failures and dynamic job arrivals, and further evaluate the algorithm's robustness and adaptability under such conditions.

Declarations

Conflict of interest

The authors declare no competing interests

Author Contribution

J. W. and P. S. completed the algorithmic principles while writing the main parts of the manuscript. Also, J. W. and P. S. completed the experiments and computed the data for the manuscript. Z. G. checked the format of the manuscript. All authors reviewed the manuscript.

Acknowledgement

Supported by Natural Science Foundation of Shaanxi Province (No.2025JC-YBMS-736, 2025JC-YBMS-343), and the Shaanxi Province Key Research and Development Project (No.2025CY-YBXM-061).

Data availability

No datasets were generated or analyzed during the current study.

References

1. Brucker P, Schlie R (1990) Job-shop scheduling with multi-purpose machines[J]. Computing 45:369–375
2. Brandimarte P (1993) Routing and scheduling in a flexible job shop by tabu search[J]. Ann Oper Res 41:157–183
3. Kacem I, Hammadi S, Borne P, Part C (2002) 32(1): 1–13
4. Xia W, Wu Z (2005) An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems[J]. Comput Ind Eng 48(2):409–425

5. Zhang G, Gao L, Li P et al (2009) Improved genetic algorithm for the flexible job-shop scheduling problem[J]. J Mech Eng 45(7):145–151 (in Chinese)
6. Li H, Li J, Du X et al (2024) IGWO algorithm for solving multi-objective flexible job shop[J]. Manuf Technol Mach Tool, (10): 174–180 (in Chinese).
7. Yu F, Xu Y, Fan R et al (2025) Research on multi-objective flexible job-shop scheduling based on an improved sparrow search algorithm[J]. Mod Manuf Eng, (12): 27–34, 104 (in Chinese).
8. Pezzella F, Morganti G, Ciaschetti G (2008) A genetic algorithm for the flexible job-shop scheduling problem[J]. Comput Oper Res 35(10):3202–3212
9. Gao J, Sun L, Gen M (2008) A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems[J]. Comput Oper Res 35(9):2892–2907
10. Zhang G, Shao X, Li P et al (2009) An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem[J]. Comput Ind Eng 56(4):1309–1318
11. Lei D, Li M, Wang L (2019) A two-phase meta-heuristic for multiobjective flexible job shop scheduling problem with total energy consumption threshold[J]. IEEE Trans Cybernetics 49(3):1097–1109
12. Gao K, Cao Z, Zhang L et al (2019) A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems[J]. IEEE/CAA J Automatica Sinica 6(4):904–916
13. Xiao Y, Cui H, Abu Khurma R et al (2025) Artificial lemming algorithm: a novel bionic meta-heuristic technique for solving real-world engineering optimization problems[J]. Artif Intell Rev 58:84
14. Bean JC (1994) Genetic algorithms and random keys for sequencing and optimization[J]. ORSA J Comput 6(2):154–160
15. Deb K, Pratap A, Agarwal S et al (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Trans Evol Comput 6(2):182–197
16. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach[J]. IEEE Trans Evol Comput 3(4):257–271
17. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer[J]. Adv Eng Softw 69:46–61
18. Mirjalili S, Saremi S, Mirjalili SM et al (2016) Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization[J]. Expert Syst Appl 47:106–119
19. Tanyildizi E, Demir G (2017) Golden sine algorithm: a novel math-inspired algorithm[J]. Adv Electr Comput Eng 17(2):71–78
20. Tizhoosh HR (2005) Opposition-based learning: a new scheme for machine intelligence[C]//Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation. 1: 695–701
21. Zhao W, Wang L, Mirjalili S (2022) Artificial hummingbird algorithm: a new bio-inspired optimizer with its engineering applications[J]. Comput Methods Appl Mech Eng 388:114194
22. Faramarzi A, Heidarinejad M, Mirjalili S et al (2020) Marine predators algorithm: a nature-inspired metaheuristic[J]. Expert Syst Appl 152:113377
23. Xue J, Shen B (2020) A novel swarm intelligence optimization approach: sparrow search algorithm[J]. Syst Sci Control Eng 8(1):22–34
24. Zitzler E, Thiele L, Laumanns M et al (2003) Performance assessment of multiobjective optimizers: an analysis and review[J]. IEEE Trans Evol Comput 7(2):117–132
25. Emmerich MTM, Deutz AH (2018) A tutorial on multiobjective optimization: fundamentals and evolutionary methods[J]. Nat Comput 17:585–609
26. Riquelme N, von Lücken C, Barán B (2015) Performance metrics in multi-objective optimization[C]//Proceedings of the 2015 Latin American Computing Conference (CLEI). : 1–11

Figures

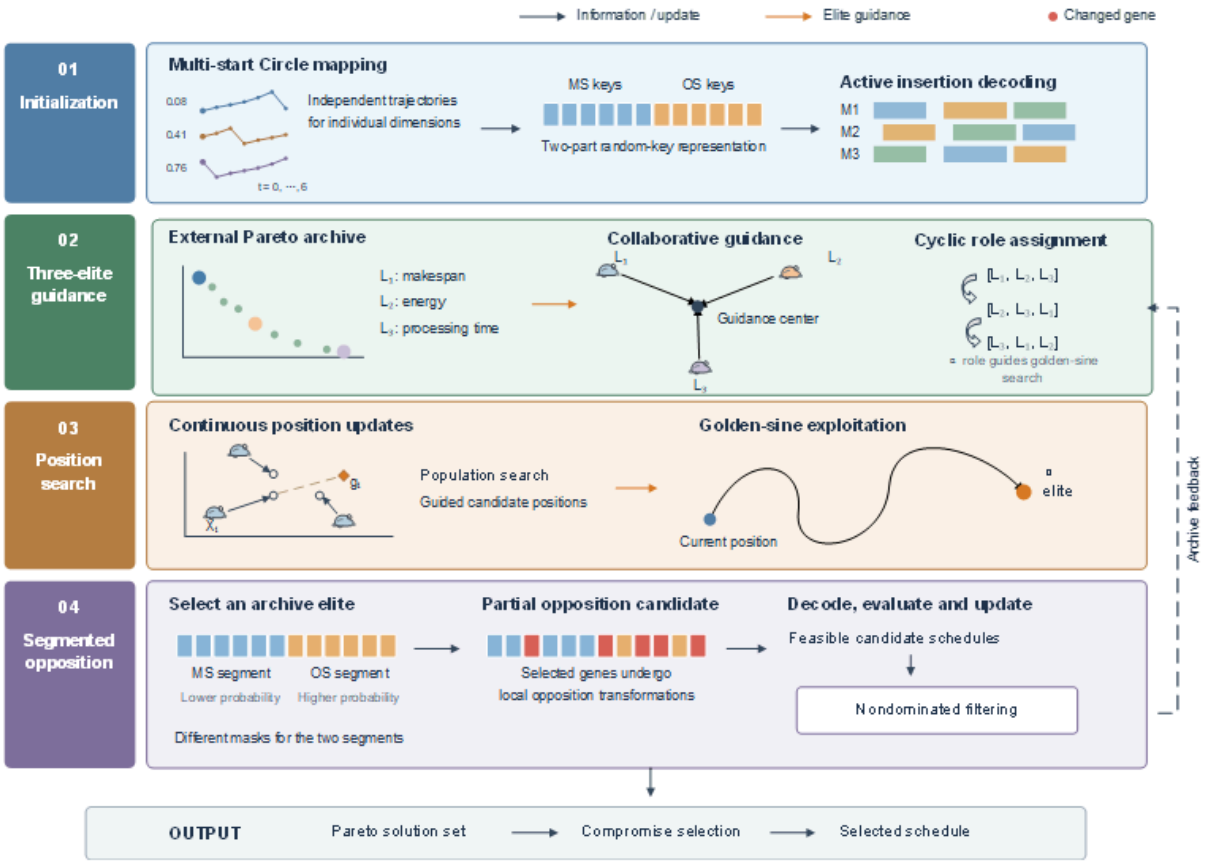
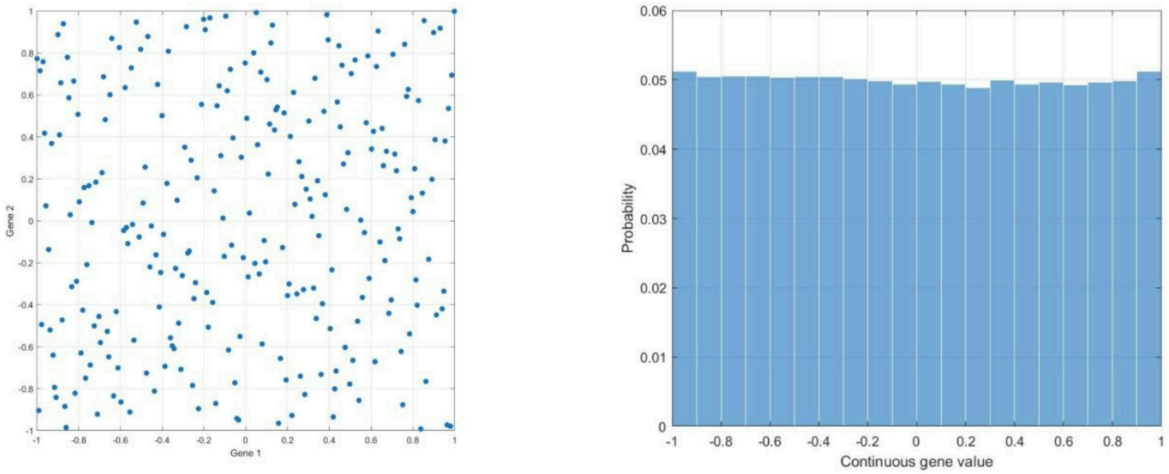


Figure 1

Algorithm Flowchart



(a) (b)

Figure 2

Distribution of the initial population generated by multi-start Circle chaotic mapping.

(a). Scatter plot of the initial solutions generated by multi-start Circle chaotic mapping.

(b). Histogram of the initial solutions generated by multi-start Circle chaotic mapping.

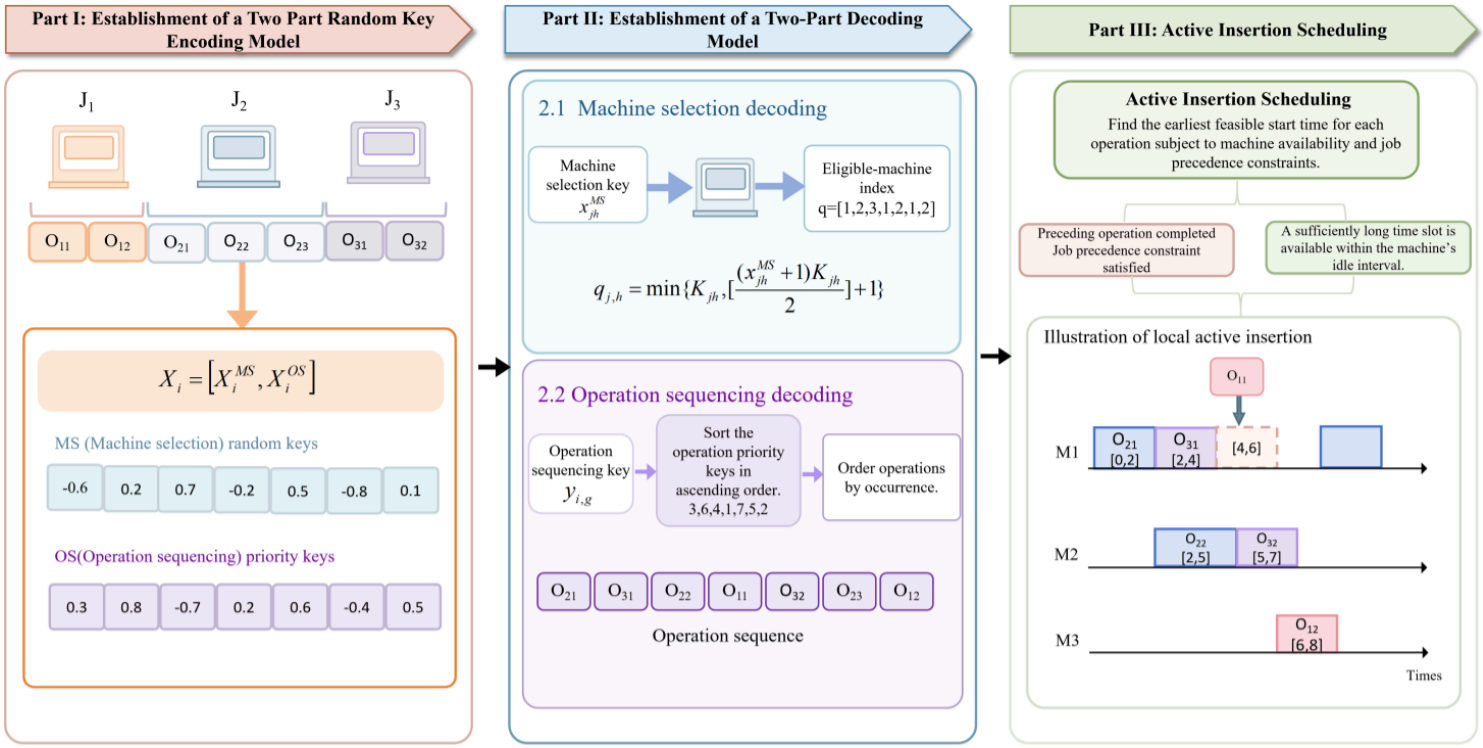


Figure 3

Schematic of the encoding and decoding procedure.

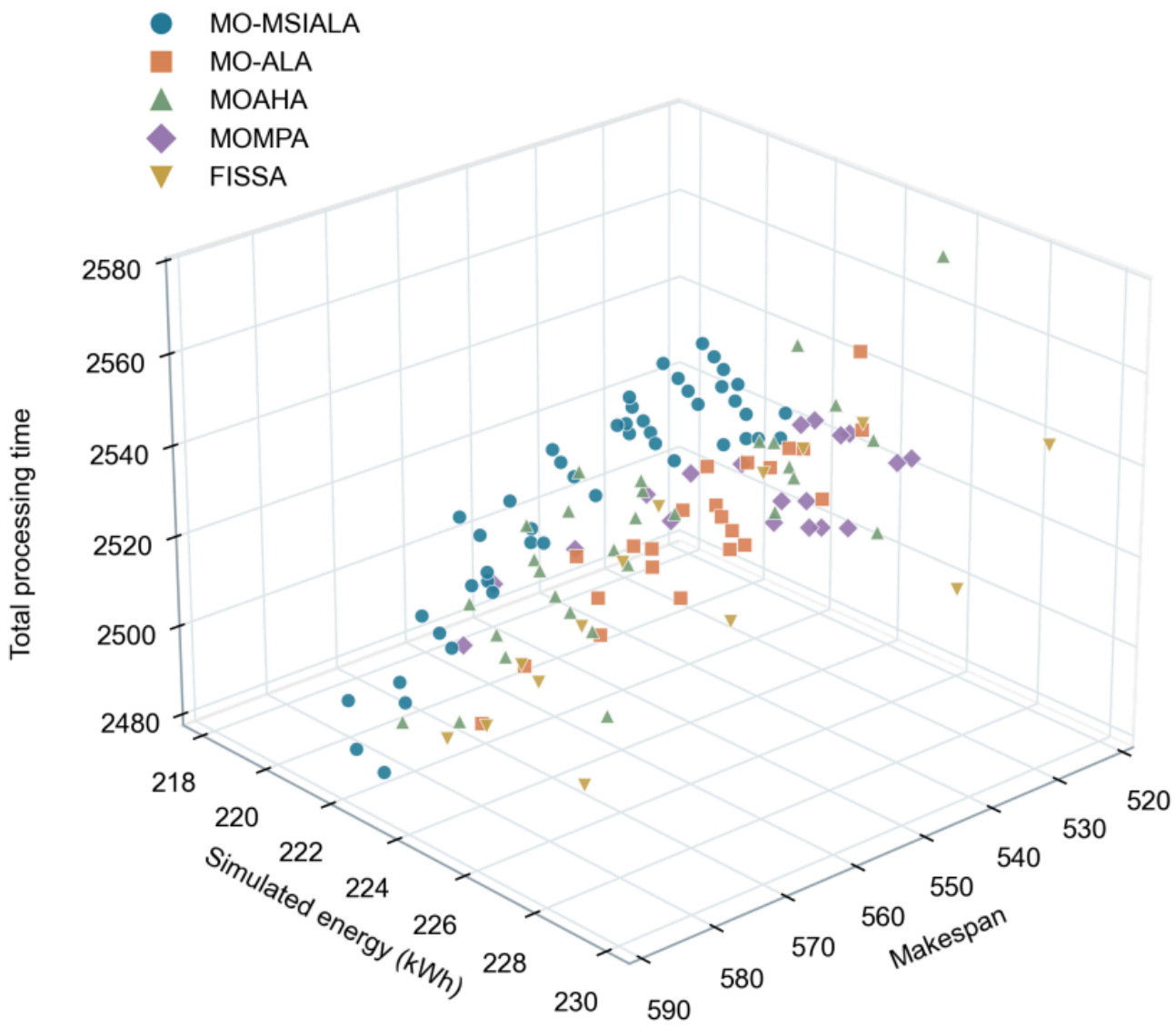


Figure 4

Three-dimensional distribution of the nondominated solution sets

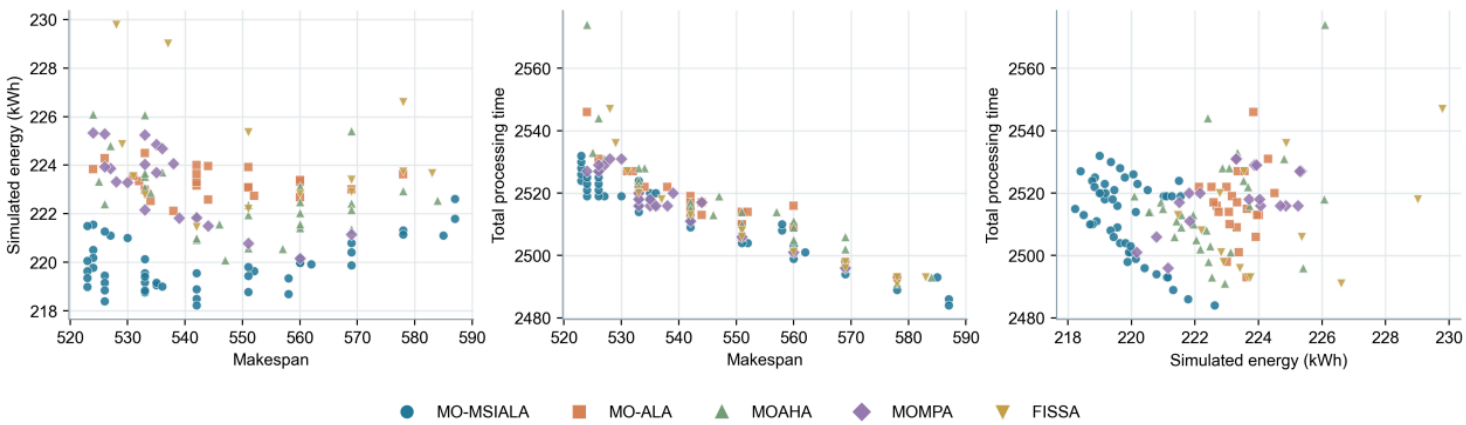


Figure 5

Two-dimensional projections of the nondominated solution sets

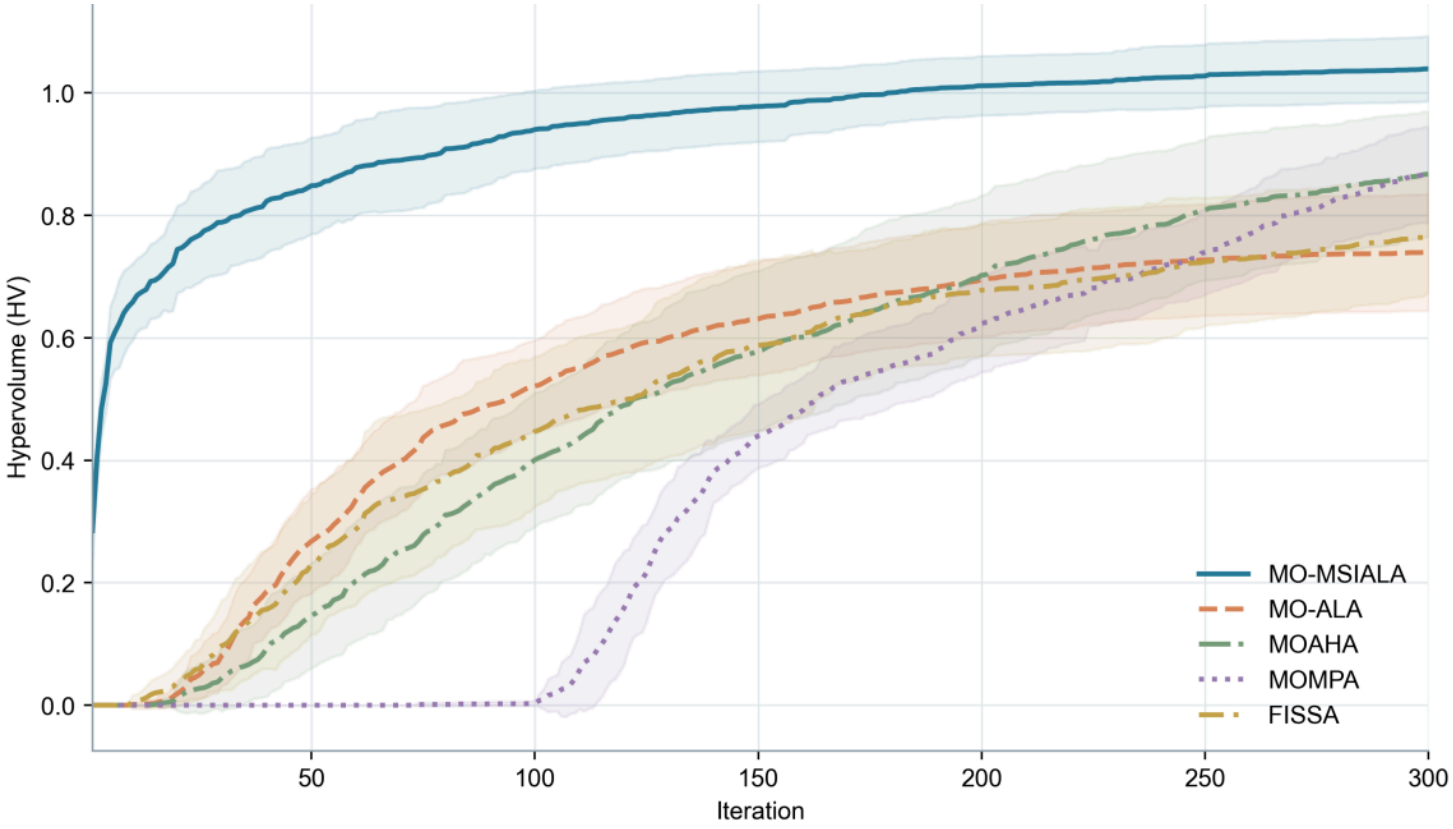


Figure 6

Mean HV convergence curves of the five algorithms on the Mk08 instance

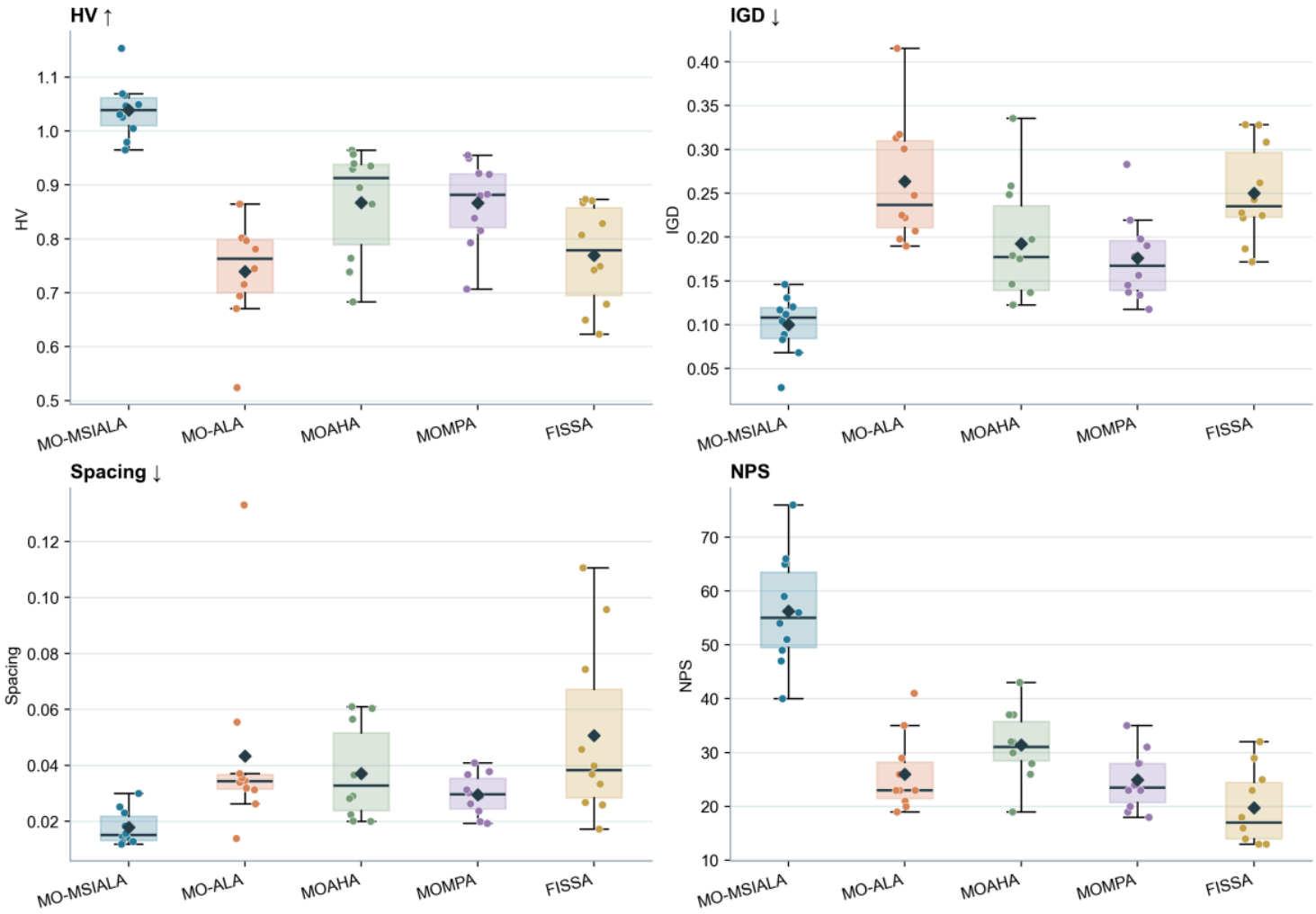


Figure 7

Distributions of the results from ten independent runs of the five algorithms on Mk08

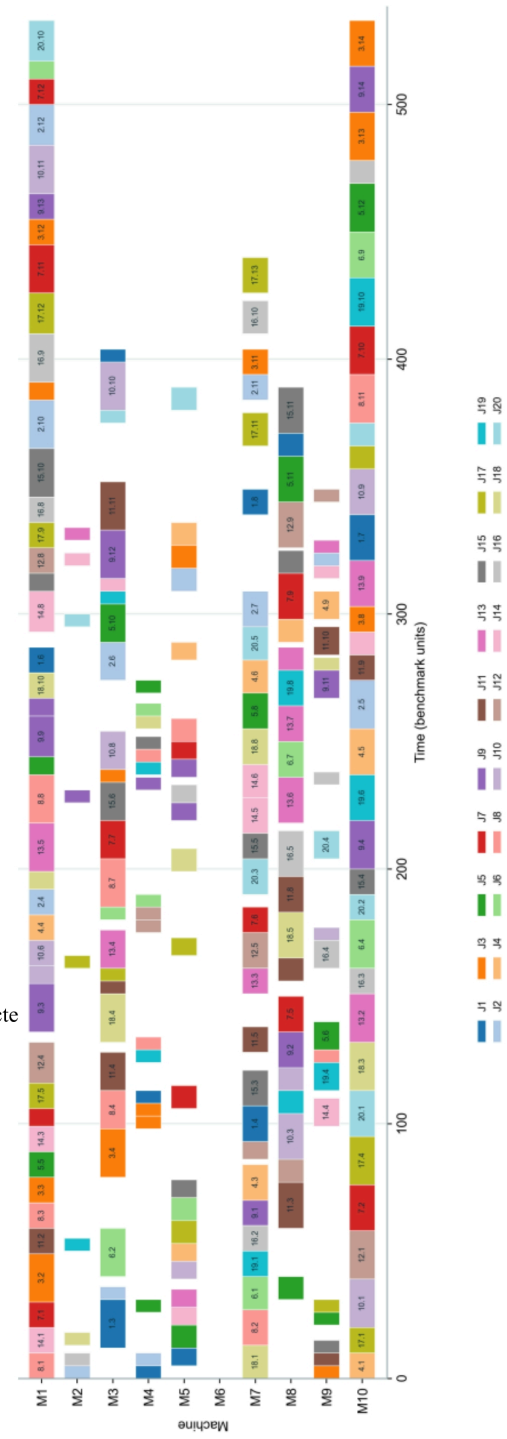


Fig. 8 Gantt chart of the selecte

Figure 8

Gantt chart of the selected compromise schedule