Research on Engineering Structures & Materials
www.jresm.org
Research Group

Research Article

Improved SA algorithm for masonry layout optimization of building infill walls

Yuanzhe Chen [*,1,2,a], Feifei Chen [3,b]

[1]*School of Science and Engineering, The Open University of China, Beijing, 100039, China*
[2]*Shandan County Comprehensive Urban and Rural Development Center, Zhangye, 734000, China*
[3]*Institute of Smart City and Intelligent Transportation (Institute of Urban Rail Transportation), Southwest Jiaotong University, Chengdu, 611756, China*

Article Info

Article History:

Received 09 June 2026
Accepted 16 July 2026

Keywords:

Building infill wall masonry;
Layout optimization;
GA;
SA;
Hybrid optimization

Abstract

This paper proposes a hybrid optimization algorithm that fuses multiple methods to address the weak global exploration ability, frequent local optima, and poor engineering adaptability in masonry layout optimization of building infill walls. The method builds a multilayer cooperative framework. It first uses the Genetic Algorithm to create a diverse population. It then applies Simulated Annealing to perform probabilistic jumping optimization. After that, it introduces sparse A search to verify topological feasibility. It finally relies on a cooperative mechanism of Adaptive Whale Optimization and iterative local search to explore the solution space in depth. Experiments on the simultaneous localization and mapping–building information modeling coupled dataset and the building information modeling component multimodal dataset show that the algorithm reaches a standard block utilization rate of 98.76 percent. It also keeps the cutting loss rate as low as 2.79%and achieves a peak stagger-joint compliance rate of 97.11%. In irregular wall scenarios, it reduces cost by up to 33.87%. The results show that this algorithm improves the optimization quality and engineering applicability of masonry layout and provides reliable technical support for precise construction and efficient material use of building infill walls.

© 2026 MIM Research Group. All rights reserved.

1. Introduction

The accuracy of masonry layout for building infill walls directly determines construction efficiency and material use. Intelligent optimization algorithms create coordinated multi objective optimization and offer new solutions for accurate layout [1]. However, traditional layout optimization algorithms perform poorly in material utilization, construction feasibility, and structural stability when the layout must satisfy complex structural constraints [2]. Accurate and intelligent masonry layout of building infill walls therefore plays an important role in improving construction quality and reducing engineering cost. The hybrid Genetic Algorithm and Simulated Annealing algorithm (GA-SA) combines population evolution strategies with probabilistic jumping mechanisms [3–4]. But existing GA-SA methods still rely on prior knowledge for parameter adjustment. They also show weak search efficiency in high-dimensional solution spaces and limited adaptability in dynamic construction scenarios [5]. This study introduces a topological verification mechanism based on Sparse A Search (SAS). It also combines the intelligent search ability of the Adaptive Whale Optimization Algorithm (AWOA) and the fine-grained exploration of Iterative Local Search (ILS). The method builds a multilayer cooperative layout optimization framework. It aims to improve the overall performance of masonry layout through coordinated global exploration, local optimization, and engineering feasibility verification. This framework provides

*Corresponding author: YuanzheyChen@outlook.com
[a]orcid.org/0009-0006-1253-8395; [b]orcid.org/0009-0009-5539-1831
DOI: http://dx.doi.org/10.17515/resm2026-1734st0609rs

technical support for green construction and industrialized building development. The innovation of this study lies in building a layout optimization model that integrates multiple intelligent algorithms through multilayer cooperation. The model applies the coordinated mechanism of GA and SA. It also merges the adaptive mechanism of AWOA with the search characteristics of ILS. The method improves engineering adaptability and solution efficiency while ensuring layout quality. It offers an innovative solution for precise construction and efficient material use of building infill walls.

1.1 Literature Review

As a classical global optimization algorithm, the SA algorithm has been widely applied in various fields, and its intelligent jump mechanism inspired by the thermodynamic annealing process has received continuous attention and research from scholars at home and abroad. For example, Gangadevi E et al. proposed a recognition method based on SA-optimized support vector machines to address the problem of crop diseases. This method combined SA with the fruit fly optimization algorithm to avoid global optimum, and then performed classification to accurately identify tomato plant diseases [6]. Kosanoglu F's team proposed a hybrid algorithm based on SA algorithm and deep reinforcement learning to prevent asset failures. This algorithm used deep reinforcement learning to find the optimal solution, and then used this solution as the initial solution of the SA algorithm to find the final optimal solution more quickly [7]. Pashaei E et al. proposed a gene selection strategy based on SA and binary coot optimization algorithm for better gene selection. This strategy used SA optimization to improve the coot algorithm to find stable information genes [8]. Lim K C W's team proposed a hyper-heuristic method based on SA algorithm for flexible job shop scheduling. This method used two variants of SA hyper-heuristics to conduct investigations based on machine assignment and job sequencing rules for job shop scheduling [9]. Zanazzo E et al. proposed a local search technique based on SA for medical student scheduling. This technique used SA to guide the combination of two different neighborhood relationships for medical student scheduling [10].

As a core component of building construction quality and efficiency, the masonry layout of building infill walls directly relates to wall structural safety, material utilization efficiency, and engineering progress control. Many domestic and foreign scholars have conducted in-depth studies on this topic. For example, Valentino J's team proposed a method based on discontinuity layout optimization and limit analysis for wall masonry layout optimization. This method used the smeared continuum approach to model the masonry, and then used limit analysis to determine the load-bearing capacity of the masonry for wall masonry layout optimization [11]. Vu Hong Son P et al. proposed a layout optimization method based on hybrid ant lion optimization algorithm to address the problem that optimization programs were only suitable for small-scale problems. This method used the ant lion optimization algorithm to avoid local optimum, and then found the optimal solution through this algorithm for effective layout optimization [12]. Tan Y's team proposed a courtyard division method based on concrete components to address the contradiction between concrete components and construction site area. This method used rectangular algorithms to calculate the number of stacks, and then used non-dominated sorting genetic algorithm for simulation to obtain better layout schemes [13]. Shang D et al. proposed an architectural structure layout optimization method based on image segmentation to address the problem of numerous constraints in building structures. This method used semantic simultaneous localization and mapping framework to achieve real-time integration, and then constructed a grid graph-based optimization model for architectural structure layout optimization [14]. Zheng Z et al. proposed a layout optimization method based on multi-population genetic algorithm for building module layout optimization. This method used multi-population genetic algorithm and typical module units to establish an integrated framework for modular construction building module layout and optimization for automated layout [15].

Based on the above content, although the existing research has made certain progress in the field of building infill wall masonry layout, it mostly focuses on macroscopic structural optimization or the application of specific algorithms, and there is still a lack of systematic research at the microscopic level on the coordinated optimization of details in masonry layout, the diversity of

masonry block specifications and the feasibility of construction. Specifically, the existing methods lack an optimization strategy that can comprehensively consider these microscopic factors when dealing with the flexibility of masonry block arrangement, the diversity of masonry block specifications, and the balance between construction efficiency and material utilization. Therefore, this study proposes a building infill wall masonry layout method based on multi-objective collaborative optimization, aiming to address the limitations of existing methods in the handling of details and the diversity of masonry block specifications. The study proposes a multi-layer collaborative optimization architecture that integrates the mechanisms of GA and SA, the rapid search ability of SAS, and the advantages of AWOA and ILS to construct an intelligent optimization algorithm for building infill wall masonry layout. Regarding the contribution of the research method to the current research gap, it is (1) proposing a multi-objective optimization model for masonry layout based on the integration of GA and SA, which can simultaneously consider the flexibility of masonry block arrangement, the diversity of masonry block specifications, and the balance between construction efficiency and material utilization. By introducing an adaptive weight adjustment mechanism, it achieves dynamic trade-offs among multiple objectives, thereby improving the comprehensive performance of the layout scheme. (2) Designing a hybrid optimization strategy based on the combination of SAS rapid search and AWOA-ILS local refinement, which uses the rapid search ability of SAS to quickly locate high-quality solution spaces and then uses the local refinement ability of AWOA-ILS to deeply explore the solution space, thereby improving the quality of the solution while maintaining search efficiency. Compared with existing optimization methods, the research method has significant advantages in the flexibility of masonry block arrangement and construction feasibility, and can effectively handle the masonry block adaptation issues at the detail nodes.

2. Masonry Layout Algorithm Based on Improved SA For Building Infill Walls

2.1 Masonry layout algorithm based on improved SA for building infill walls

In the intelligent optimization of masonry layout for building infill walls, accurate construction and green building require coordinated computation between layout pattern generation and intelligent optimization algorithms. This process builds a complete solution from data perception to decision output [16]. The SA algorithm uses its strong global exploration ability to search the optimal or near-optimal masonry layout [17]. The workflow of SA for masonry layout optimization is shown in Figure 1.

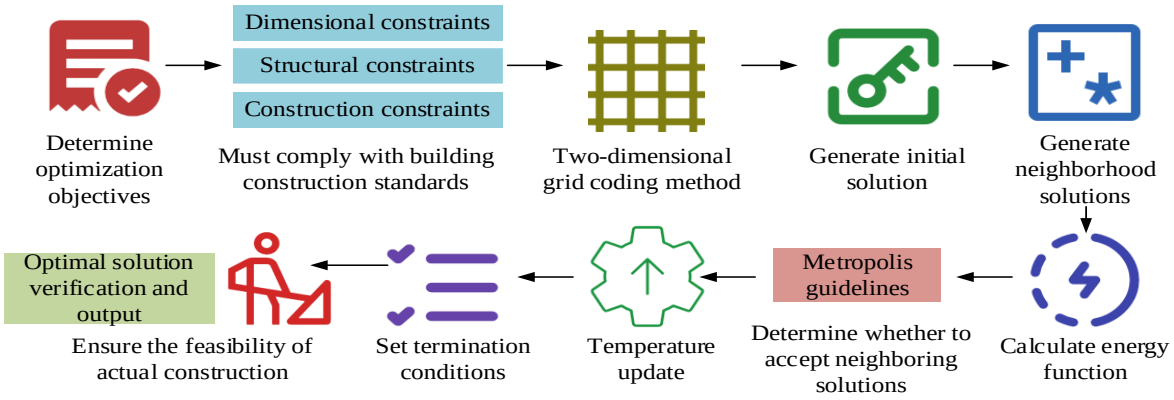


Fig. 1. Flowchart for SA to optimize the layout of masonry infill walls in buildings

According to Figure 1, SA first sets the optimization objectives. The objectives are minimum block loss and maximum construction efficiency under multiple constraints. It then generates an initial solution that satisfies the constraints based on standard construction logic. After that, the algorithm enters the core annealing iteration. It adjusts the current solution to create a neighborhood solution. It evaluates the solution with the energy function. It decides whether to update the solution based on the acceptance rule. It lowers the temperature based on the fast-to-slow schedule. The algorithm stops when the temperature reaches the threshold, the energy does not change across iterations, or the maximum iteration number is reached. It then verifies engineering

feasibility and outputs the optimal solution. The initial temperature in the SA parameter is set to 1000, the cooling rate is set to 0.95, and the termination temperature is set to 1×10^{-3}. At each temperature, the length of the Markov chain is set to 100, which means conducting 100 neighborhood searches at each temperature. The design of the energy function comprehensively considers three dimensions: block utilization rate, construction convenience, and structural safety. The weight of block utilization rate is set to 0.5, the weight of construction convenience is set to 0.3, and the weight of structural safety is set to 0.2. The parameter selection method is based on a combination of empirical trial and error and sensitivity analysis. Firstly, parameter pre tuning is carried out through small-scale test cases, and then the orthogonal experimental design method is used to systematically screen the key parameter combinations, ultimately determining the parameter configuration that balances the convergence speed and solution quality of the algorithm. The temperature update of SA is shown in Equation (1).

$$T_{k+1} = \alpha \cdot T_k \tag{1}$$

In Equation (1), T_k represents the temperature of the k -th iteration. T_{k+1} represents the temperature of the $k+1$ -th iteration. α represents the temperature coefficient. The state transition probability of SA is shown in Equation (2).

$$\begin{cases} P = 1 & E(s') \le E(s) \\ P = e^{\frac{E(s')-E(s)}{T_k}} & E(s') > E(s) \end{cases} \tag{2}$$

In Equation (2), $E(s)$ represents the energy of the current state. $E(s')$ represents the energy of the new state. P represents the transition probability. s and s' represent the current state of the algorithm and the new state generated by the algorithm. But standard SA shows a single search direction in complex solution spaces and suffers from early convergence [18]. This study therefore introduces GA into SA to form a hybrid algorithm that enhances global optimization performance. GA conducts preliminary screening of population size, crossover probability, and mutation probability through pre-experiments to determine their reasonable range of values. On this basis, orthogonal experimental design was adopted to finely tune the above parameters, with the weighted sum of population diversity and convergence speed as the evaluation index, and the optimal parameter combination of GA was finally determined. The workflow of GA optimizing SA is shown in Figure 2.

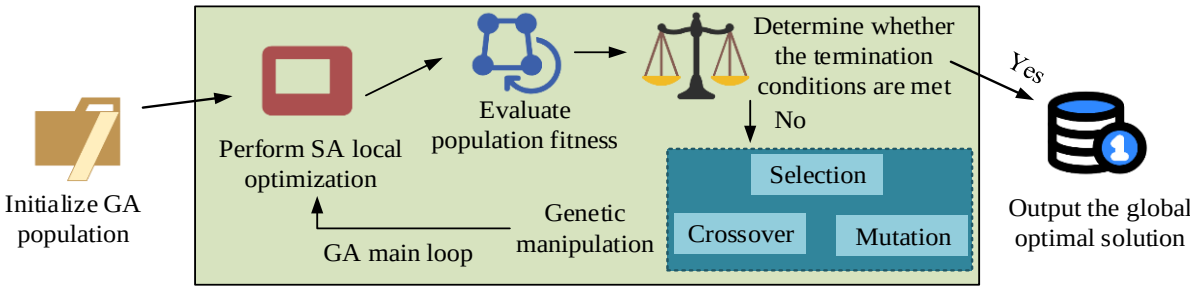


Fig. 2. Flowchart of GA optimizing SA

According to Figure 2, GA first creates an initial population that represents different masonry layout schemes. Each individual then enters the SA local optimization stage. The probabilistic jumping mechanism improves the quality of each individual. The optimized population then undergoes genetic operations. The new population enters another SA local optimization process. The cooperation between the global exploration of GA and the local search of SA outputs the optimal masonry layout for building infill walls. The layout of the masonry structure adopts a binary encoding method, mapping the arrangement scheme of the masonry blocks into chromosome individuals. Each gene position corresponds to the position status of a masonry block,

with 0 indicating that no masonry block is placed at that position and 1 indicating that a masonry block is placed. Assuming that the horizontal direction of the wall can accommodate 15 bricks and the vertical direction can accommodate 24 layers, the encoding length is 360. The selection function of GA is shown in Equation (3).

$$\begin{cases} p_i(t) = \dfrac{f\left(x_i(t)\right)}{F(t)} \\ \sum_{j=1}^{k-1} p_j(t) \le r < \sum_{j=1}^{k} p_j(t) \end{cases} \tag{3}$$

In Equation (3), $p_i(t)$ represents the selection probability of the t -th individual in the i -th generation. $f\left(x_i(t)\right)$ represents the individual fitness. $F(t)$ represents the sum of all fitness values. $\sum_{j=1}^{k-1} p_j(t)$ and $\sum_{j=1}^{k} p_j(t)$ represent the cumulative selection probabilities of the first $k-1$ individuals and the first k individuals. The crossover function is shown in Equation (4).

$$x_{a'}(t+1) = (a_1, \cdots, a_c, b_{c+1}, \cdots, b_L) x_{b'}(t+1) = b_1, \cdots b_c, a_{c+1}, \cdots a_L) \tag{4}$$

In Equation (4), $x_{a'}(t+1)$ represents the gene sequence of the parent individual in the next generation. c represents the random crossover point. L represents the gene length. The mutation function is shown in Equation (5).

$$g_j' = \begin{cases} 1 - g_j & if\ r < p_m \\ g_j & otherwise \end{cases} \tag{5}$$

In Equation (5), g_j' is the value at the j -th gene position, and r is a random number. p_m represents the mutation probability. Although GA-SA enhances layout optimization by combining population-based search and probabilistic jumps, it still shows limited search efficiency in large solution spaces. The SAS algorithm improves efficiency through sparse state sampling and adaptive heuristic evaluation. It builds an effective layout path through multi-constraint topological spaces [19]. Compared to traditional A * algorithms, SAS exhibits stronger global optimization capabilities in masonry layout, especially suitable for complex layout problems under multiple constraint conditions. The core mechanism is to transform the masonry layout problem into a path search problem in sparse state space, dynamically evaluate the priority of each state node through heuristic functions, and use adaptive sampling strategies to balance exploration and utilization. Therefore, the study introduces the SAS algorithm into the GA-SA algorithm to construct an optimization algorithm (GA-SA-SAS, abbreviated as GSSA) to improve search efficiency. The workflow of GSSA for masonry layout optimization is shown in Figure 3.

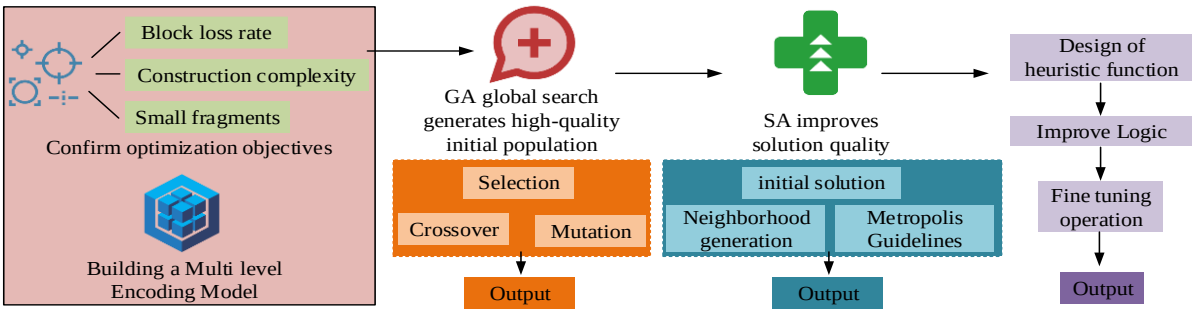


Fig. 3. Flowchart of GSSA for masonry layout optimization

According to Figure 3, the method first sets the objectives as minimum block loss and minimum construction complexity. It defines constraints such as mortar thickness and cutting blocks. It builds a two-dimensional grid and sequence-encoding hybrid model. GA then performs global

search. It creates an initial population, executes genetic operations, and outputs high-quality individuals. These individuals then enter SA as initial solutions. SA performs neighborhood search and probabilistic acceptance under the acceptance rule. It lowers the temperature based on the cooling coefficient and outputs the minimum-energy solution. SAS then uses its heuristic function to optimize the block placement path and sequence. It solves construction adaptation problems and stops when the termination condition is satisfied. The algorithm outputs the optimal layout. The heuristic function in SAS dynamically guides the search direction by evaluating the sum of the estimated cost from the current node to the target node and the consumed cost in a sparse state space, thereby prioritizing paths with lower construction complexity and higher material utilization in masonry layout. This function integrates block dimensions, mortar joint constraints, and cutting limitations, adaptively adjusting the weights of each factor to balance global optimality and local feasibility in the search process. The study sets adaptive weights based on block size deviations and mortar joint thickness fluctuations, dynamically optimizing the exploration-exploitation balance coefficient through real-time feedback, thereby refining the path selection strategy in masonry layout.

During the execution of construction constraints, the study transforms block dimensions, mortar joint thickness, and cutting limitations into constraint conditions through encoding rules, dynamically checking the feasibility of each state node during the search process. If a current node violates constraints, its priority is reduced via a penalty function, guiding the search toward feasible regions. Additionally, a backtracking mechanism records historical states, automatically reverting to the last feasible node and reselecting branch paths when the search repeatedly falls into local optima, avoiding invalid iterations. The study introduces a dynamic constraint relaxation strategy, relaxing mortar joint thickness or cutting block quantity limits when construction material utilization is low to expand the feasible solution space. When material utilization falls below 85%, the relaxation mechanism is triggered, allowing the permissible deviation of mortar joint thickness to increase from ±2mm to ±3mm, while also permitting a 10% increase in the upper limit of cutting blocks. The total cost of SAS is shown in Equation (6).

$$f(n) = g(n) + h(n) \tag{6}$$

In Equation (6), $f(n)$ represents the total cost of the current placement state. $g(n)$ represents the actual cost from the initial unplaced state to state n. $h(n)$ represents the estimated minimum cost from state n to the fully placed state. To ensure the feasibility of the solution, the model incorporates structural safety redundancy coefficients and construction tolerance ranges in the constraints, enabling the optimization results to meet the specification requirements while also having the operational feasibility and fault-tolerant capability in actual engineering. Additionally, during the optimization solution process, each iteration strictly verifies the constraints. If a solution violates any constraint, it will be automatically eliminated and a new candidate solution will be generated to ensure that the final output solution fully complies with all constraints. Moreover, the model also introduces a penalty function mechanism, applying penalty terms to the candidate solutions that violate the constraints, guiding the search direction towards the feasible region.

2.2 Masonry Layout Algorithm Based on GSSA

GSSA coordinates global exploration and local optimization in masonry layout optimization for building infill walls. But the three-layer coupling of the algorithm still causes high computational cost. AWOA introduces dynamic adaptive weights and a nonlinear convergence factor. It balances the spiral encircling behavior and random hunting behavior. It maintains population diversity and accelerates convergence. It therefore improves the generation efficiency and stability of masonry layout in engineering environments [20]. The structure of AWOA for masonry layout optimization is shown in Figure 4.

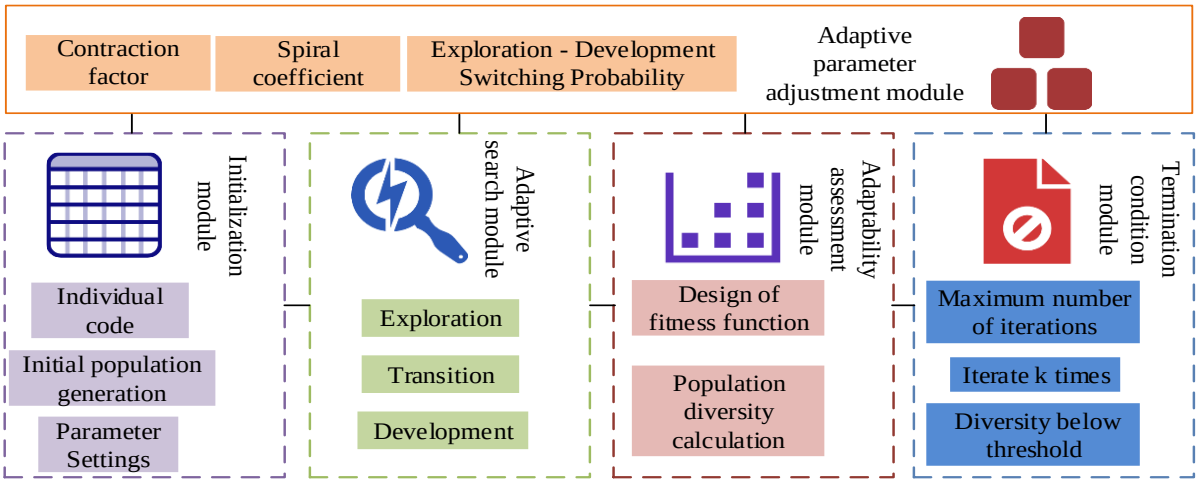


Fig. 4. Structure of AWOA

According to Figure 4, AWOA contains four core components. In the population initialization module, each whale individual uses a position vector to represent a complete masonry layout. The core improvement module uses an adaptive mechanism to balance global exploration and local exploitation. The position update module contains three search modes inspired by whale hunting behavior. The evaluation and iteration module evaluates the quality of solutions with the fitness function and uses an elite retention strategy to keep the optimization direction correct. The encircling behavior of AWOA is shown in Equation (7).

$$X(t+1) = X(t) + r(X^* - X(t)) \tag{7}$$

In Equation (7), $X(t)$ and X^* represent the current position and the optimal position. r represents a random number in the range [-1, 1]. The hunting behavior of AWOA is shown in Equation (8).

$$\begin{cases} X(t+1) = X^* - A \cdot D_1 \cdot e^{bl} \cdot \cos(2\pi l) \\ D_1 = \left| C \cdot X^* - X(t) \right| \end{cases} \tag{8}$$

In Equation (8), D_1 represents the distance between the current data and the optimal data. A and C represent coefficient vectors. b is a constant that controls the spiral shape. l is a random number in the range [-1, 1]. The prey search behavior of AWOA is shown in Equation (9).

$$X(t+1) = X_{rand} - A \cdot D_2 \cdot e^{bl} \cdot \cos(2\pi l) \tag{9}$$

In Equation (9), D_2 represents the distance between the current data and a random data point. X_{rand} represents the position of a randomly selected data point. Because of uncertain behavior, the final update of the data position uses a random probability to determine which behavior is used. The behavior selection rule is shown in Equation (10).

$$X(t+1) = \begin{cases} X^* - A \cdot \left| C \cdot X^* - X(t) \right|, p < 0.5 \\ D_2 e^{bl} \cos(2\pi l) + X^*, p \geq 0.5 \end{cases} \tag{10}$$

In Equation (10), p represents the random probability. The algorithm performs the hunting behavior when the probability is less than 0.5. It performs the prey search behavior when the probability is greater than or equal to 0.5. The parameter update method of the AWOA algorithm adopts an adaptive adjustment strategy, dynamically adjusting the values of the coefficient vector and random probability based on the number of iterations, thus achieving a dynamic balance between global search and local development. In the early stages of iteration, larger coefficient

vectors and lower random probability thresholds are advantageous for global search to extensively explore the solution space. The study set it as an initial value of 0.9 and linearly decreased it to 0.1 with the number of iterations. At the same time, the random probability threshold gradually increased from 0.5 to 0.9, making the algorithm more inclined towards local fine development in the later stage. Traditional AWOA relies heavily on the guidance of the leader whale. This reliance may cause the algorithm to fall into a single search pattern because of population social behavior. ILS avoids this problem through multi-scale perturbation and elite restart. It prevents insufficient exploration and early convergence. This study therefore introduces ILS into AWOA to build the AWOA-ILS algorithm. The workflow of AWOA-ILS for masonry layout optimization is shown in Figure 5.

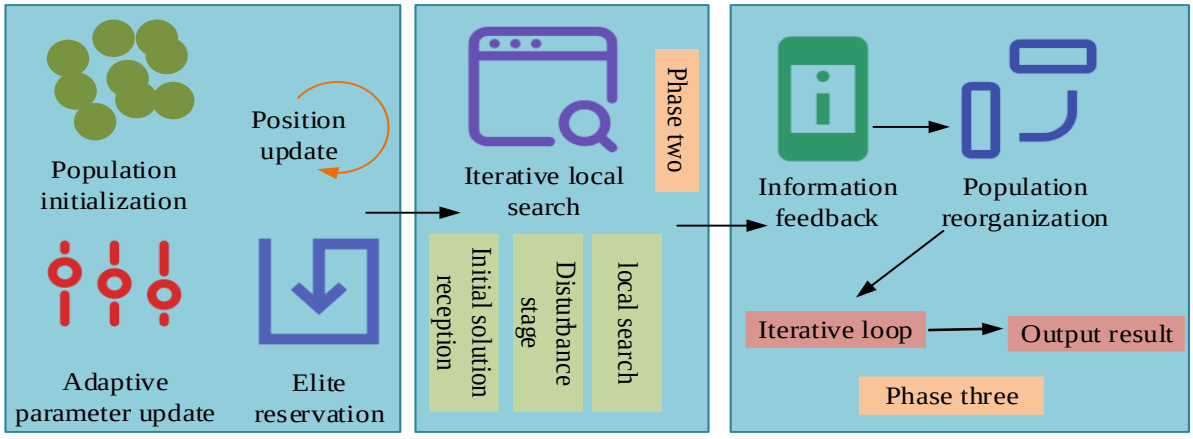


Fig. 5. AWOA-ILS's flowchart for masonry layout optimization

As shown in Figure 5, during the iterative process of the AWOA-ILS algorithm, AWOA is responsible for global exploration and population update, while the ILS module performs deep perturbation and restart of the elite solution in the local stage. In each generation iteration, AWOA first generates candidate solutions based on adaptive parameters and updates the positions of the whale population; subsequently, the ILS module performs multi-scale perturbation operations on the current optimal solution, including random exchange of block types, fine-tuning of block position coordinates, and recombination of construction sequences, to generate several variant solutions. After local search, if the fitness of these variant solutions is better than the original solution, the original optimal solution is replaced and re-injected into the population, thereby guiding the population to escape from the local optimal region. The local search expression of ILS is shown in Equation (11).

$$\cos t(s^*) = \sum_{k=1}^{n} d(v_k, v_{k+1}) \tag{11}$$

In Equation (11), $\cos t(s^*)$ represents the total cost of path s^*. n represents the total number of nodes to visit. $d(v_k, v_{k+1})$ represents the distance from node v_k to node v_{k+1}. v_k represents the k-th visited node. The perturbation and operation expression of ILS is shown in Equation (12).

$$s' = swap\left(s^*, randselect(s^*, k)\right) \tag{12}$$

In Equation (12), s' is the new solution produced after perturbation, and s^* is the current local optimal solution. k represents the perturbation strength parameter. Based on the advantages of AWOA-ILS, this study integrates AWOA-ILS with GSSA to form the AWOA-ILS-GSSA algorithm (AIGSSA). This algorithm provides a more effective optimization strategy for masonry layout of building infill walls. The workflow of AIGSSA is shown in Figure 6.

As shown in Figure 6, the AIGSSA algorithm consists of five core modules: GA, SA, SAS, AWOA, and ILS. These modules work together to complete the entire optimization process from generating the initial population to outputting the final solution. The algorithm first generates the initial masonry

arrangement population through the GA module, then the SA module performs local optimization on the population individuals, the SAS module verifies the topological feasibility, the AWOA module uses the enclosure-bubble network mechanism to mine high-quality solutions, and the ILS module implements multi-scale perturbation on the elite solutions.

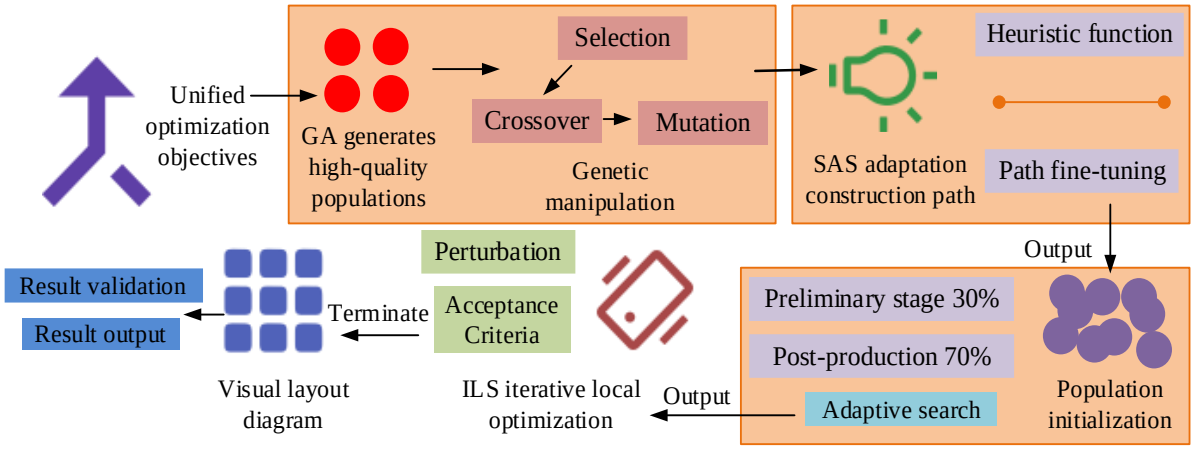


Fig. 6. AIGSSA's flowchart for masonry layout optimization

The modules form a closed-loop optimization system through the exchange of solution information and population recombination mechanisms. On the basis of ensuring the feasibility of the construction, it achieves the global optimization of the masonry arrangement scheme and the combination of engineering practicability. The optimization process maintains the diversity of the initial population through the crossover and mutation operations of the GA module, avoiding premature convergence. Subsequently, the SA module accepts inferior solutions with a certain probability through the Metropolis criterion, further enriching population diversity and preventing falling into local optima. In the SAS module, individuals with reasonable preservation structures are selected through topological constraints to ensure that the population does not deviate from the feasibility boundary while increasing diversity. The AWOA module conducts deep mining of the population through the enclosing bubble network mechanism, improving the quality of solutions while maintaining diversity. The ILS module dynamically balances local and global searches by applying multi-scale perturbations to elite solutions, avoiding population homogenization. The GA module stops when the number of iterations reaches 50 or the fitness changes by less than 0.1% for 5 consecutive generations. The SA module stops when the temperature drops to 0.01 °C. The SAS module automatically terminates after completing all individual verifications. The AWOA module stops when iterating for 30 generations or when the optimal solution has not been updated for 10 consecutive generations. The ILS module stops when iterating for 20 generations or when the elite solution has not improved for 5 consecutive generations. The optimization objective function of the AIGSSA wall masonry arrangement optimization algorithm is shown in Equation (13).

$$F(x)=\alpha\cdot f_{structural}(x)+\beta\cdot f_{efficiency}(x)+\gamma\cdot f_{cost}(x) \tag{13}$$

In Equation (13), $F(x)$ represents the optimization objective function, x is the decision variable, $f_{structural}(x)$ is the structural rationality index function, α is the weight coefficient of the structural rationality indicator, $f_{efficiency}(x)$ is the construction efficiency indicator function, β is the weight coefficient of the construction efficiency indicator, $f_{cost}(x)$ is the economic indicator function, and H is the weight coefficient of the economic indicator. The three weight coefficients are determined through the Analytic Hierarchy Process (AHP), taking 0.45, 0.35, and 0.20 respectively, to balance the priority relationship among structural safety, construction efficiency, and economy. This objective function guides the algorithm during the optimization process to prioritize improving construction efficiency while also considering material cost control, thereby achieving the maximum comprehensive benefit in engineering practice. The structural rationality indicator function is shown in Equation (14).

$$f_{structural}(x)=\sum_{i=1}^{n} w_i \cdot s_i(x) \tag{14}$$

In Equation (14), n represents the total number of structural rationality sub-indicators, i is the sequence number of the sub-indicators, w_i is the weight of the i th indicator, which is determined by expert scoring and engineering experience, $s_i(x)$ is the scoring function of the i th indicator, and the function is linearly normalized to [0,1]. The construction efficiency indicator and economic efficiency indicator functions are both in the form of multi-indicator weighted summation. The weights of the sub-indicators are also determined through expert assessment and normalized to a unified dimension. Among the sub-indicators of structural rationality, the weight of the block joint length is the highest, set at 0.50. The uniformity of the mortar joint and the overall stability of the wall are set at 0.30 and 0.20 respectively, highlighting the dominant role of the joint quality on the structural safety. In the construction efficiency indicator, the weight of the number of block types is set at 0.40, the regularity of arrangement is set at 0.35, and the operation complexity is set at 0.25, reflecting the core role of reducing the types of components in improving construction efficiency. In the economic indicators, the weight of material utilization is set at 0.50, the cost of a single block is set at 0.30, and the transportation and storage costs are set at 0.20.

This optimization problem can be modeled as a Mixed Integer Linear Programming (MILP) model, where the selection of block types and the layout schemes of blocks are integer variables, while the structural rationality, construction efficiency and economic indicators are continuous variables. The constraints include structural safety limits, material cost upper limits, block size modulus matching, and construction operability requirements. The objective function is the maximization of the weighted sum of three indicators, and its mathematical expression is shown in Equation (13). In Equation (13), α, β, and γ are the weight coefficients of the structural rationality, construction efficiency and economic indicators, satisfying α + β + γ = 1. Their values can be dynamically adjusted according to the actual engineering requirements. The optimization solution can be obtained by using heuristic algorithms such as branch and bound method or genetic algorithm, while ensuring the solution accuracy, and taking into account the computational efficiency and engineering practicability.

To conduct theoretical analysis on convergence, a convergence criterion is introduced and a threshold for the rate of change of the objective function value during the iteration process is defined. When the relative change of the objective function for three consecutive iterations is less than the preset tolerance ε, the algorithm is judged to have converged. At the same time, a multi starting point strategy is adopted to avoid getting stuck in local optimal solutions, and each starting point runs independently to select the global optimal result. When the values of α, β, and γ fluctuate within the range of [0.2, 0.5], the variation range of the structural rationality index does not exceed 8%, the variation range of the construction efficiency index does not exceed 6%, and the variation range of the economic index does not exceed 5%. This indicates that the model has good robustness with respect to the weight parameters. When the number of block types is between 3 and 5, the optimization results tend to be stable. The structural rationality index reaches above 0.85, the construction efficiency index exceeds 0.80, and the economic index remains above 0.75. When the number of block types exceeds 5, the construction efficiency index shows a significant decrease, with a drop of 12%, indicating that too many block types will increase the construction complexity and weaken the efficiency advantage. In summary, when the number of block types is 4, the comprehensive score of the three indicators reaches the optimal value, achieving a good balance among structural rationality, construction efficiency, and economy.

3. Performance Verification of The AIGSSA Masonry Layout Optimization Algorithm

3.1 Optimization Quality Analysis of the AIGSSA Algorithm

To verify whether the optimization quality of the AIGSSA wall masonry layout optimization algorithm was excellent, this study introduced three comparison algorithms for comparative

experiments with AIGSSA: the wall masonry layout optimization algorithm based on Particle Swarm Optimization and Ant Colony Optimization (PSO-ACO), the wall masonry layout optimization algorithm based on Artificial Bee Colony and Glowworm Swarm Optimization (ABC-GSO), and the wall masonry layout optimization algorithm based on Gravity Search Algorithm and Differential Evolution (GSA-DE). The experimental system used Windows 11, with an Intel i9-14900KS central processing unit and an NVIDIA RTX 4060ti graphics card. The datasets used were the SLAM-BIM Coupled Dataset (SLABIM) and the Building Information Modeling Component Multimodal Dataset (BIMCompNet). The SLABIM dataset was generated through deep integration of actual construction site point cloud data captured by simultaneous localization and mapping technology with design data from building information models, accurately reflecting the wall geometric forms, spatial relationships, and construction states in real construction site environments. The BIMCompNet dataset integrated BIM model components from various sources and types, covering walls, blocks, and associated components of different specifications, materials, and construction logic. The dataset SLABIM contains 12,000-point cloud-BIM paired samples, covering 6 common wall types and 4 construction stages; the BIMCompNet dataset contains 8,500 multi-modal component samples, covering 15 block specifications and 5 connection methods. In the experiments, the two datasets were respectively used to verify the adaptability and generalization ability of the algorithm in different scenarios. The SLABIM dataset was randomly divided into training set, validation set and test set, with a ratio of 7:1:2, to evaluate the wall layout optimization effect of the algorithm on real construction scenarios; the BIMCompNet dataset was stratified sampled by component type and divided into training set, validation set and test set in a ratio of 8:1:1 to test the robustness of the algorithm in multi-modal component layout. The original point cloud data of the SLABIM dataset was sourced from the construction site of a large commercial complex project, and was preprocessed through noise removal, registration and down sampling, and then spatially aligned and semantically labeled with the corresponding BIM model. The BIMCompNet dataset was constructed by combining the public BIM model library and self-built models, and underwent preprocessing steps such as format standardization, component classification and attribute annotation. All data were de-identified and did not involve specific project privacy information. To ensure the fairness and reproducibility of the experimental results, all experiments were conducted under the same hardware environment. Each group of experiments was repeated 5 times and the average value was taken as the final result.

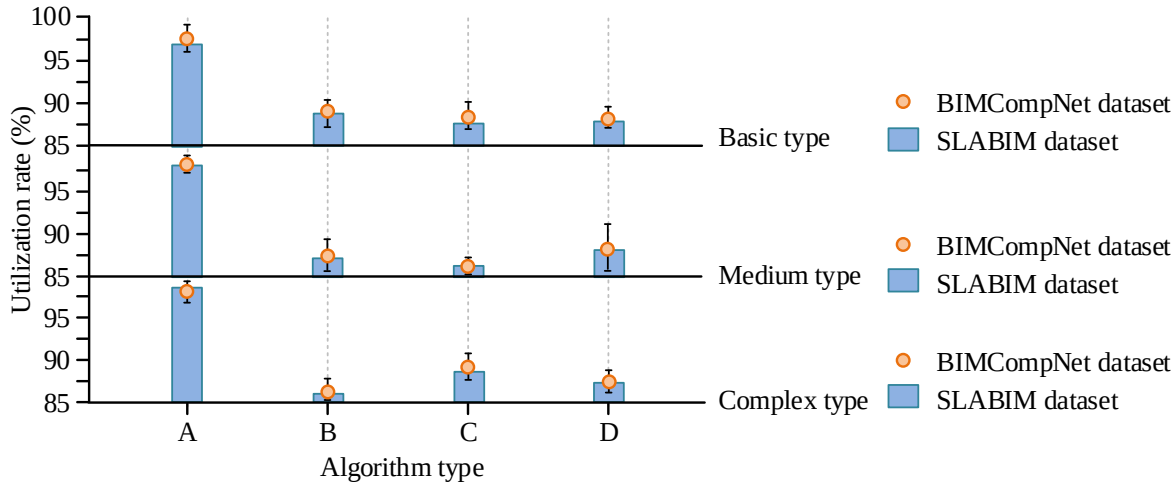


Fig. 7. Comparison of standard block utilization rate results

To verify the material control capability of the AIGSSA algorithm, this study tested the standard block utilization rate of the four algorithms for layout optimization under different wall conditions in the SLABIM and BIMCompNet datasets. The test results are shown in Figure 7. In Figure 7, A, B, C, and D represented the AIGSSA algorithm, PSO-ACO algorithm, ABC-GSO algorithm, and GSA-DE algorithm, respectively. As shown in Figure 7, in the SLABIM dataset, when the AIGSSA algorithm faced basic, medium, and complex wall conditions, its standard block utilization rates were 98.67%, 97.89%, and 98.24%, respectively. The standard block utilization rates of the other three

comparison algorithms PSO-ACO, ABC-GSO, and GSA-DE were much lower than the AIGSSA algorithm. In the BIMCompNet dataset, the standard block utilization rates of the AIGSSA algorithm were 98.14%, 98.08%, and 97.83%, respectively, and remained higher than the other three comparison algorithms. In summary, the AIGSSA wall masonry layout optimization algorithm maintained high standard block utilization rates when facing wall conditions of different complexity levels, verifying the excellent performance of the AIGSSA algorithm in optimization quality. Computational complexity refers to the time and space resources required by an algorithm during its execution, and it is an important metric for evaluating the efficiency of an algorithm. To demonstrate the superiority of the AIGSSA algorithm in material loss, this study tested the cutting loss rate of materials during layout optimization for the four algorithms in the two datasets. Cutting loss rate = (1 - Standard block usage area / Total wall area) × 100%. The test results are shown in Figure 8.

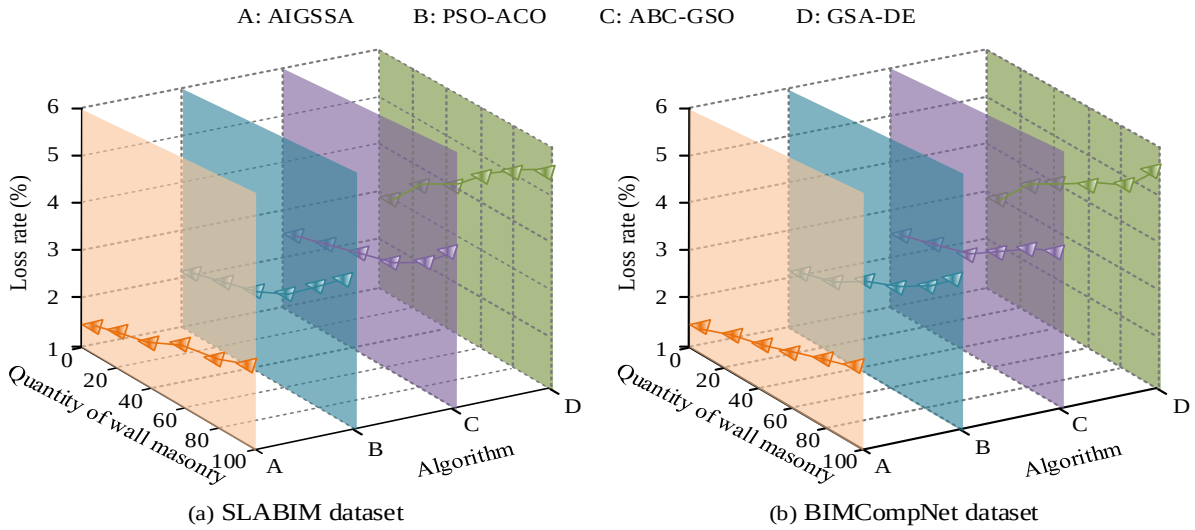
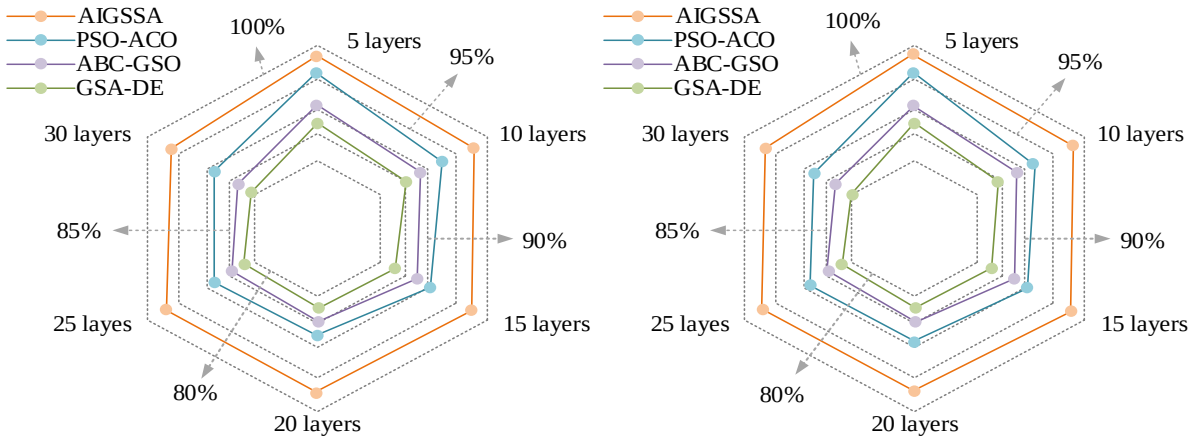


Fig. 8. Comparison of cutting loss rate test results

As shown in Figure 8(a), in the SLABIM dataset, when the number of wall masonries was 100, the cutting loss rate of the AIGSSA algorithm was 2.86%. For the other three comparison algorithms PSO-ACO, ABC-GSO, and GSA-DE, when the number of wall masonries was 100, the cutting loss rates were 4.21%, 4.33%, and 5.82%, respectively. In the BIMCompNet dataset in Figure 8(b), when the number of wall masonries reached 100, the cutting loss rate of the AIGSSA algorithm further decreased to 2.79%. In summary, the AIGSSA algorithm, with its unique optimization mechanism and efficient search strategy, could more accurately find the optimal solution during the wall masonry layout optimization process, thereby effectively reducing the cutting loss rate. Subsequently, to verify the rationality of the masonry sequence of the AIGSSA algorithm, this study tested the joint staggering compliance rate of the four algorithms during layout optimization in the above two datasets. The calculation formula for the staggered joint compliance rate is as follows: Staggered joint compliance rate = (Number of blocks meeting the staggered joint requirements / Total number of blocks) × 100%. The test results are shown in Figure 9.

According to Figure 9(a), in the SLABIM dataset, when the masonry layer data volume was 30 layers, the joint staggering compliance rate of the AIGSSA algorithm was 96.13%. The joint staggering compliance rates of the other three comparison algorithms PSO-ACO, ABC-GSO, and GSA-DE were 89.25%, 84.33%, and 82.14%, respectively. As shown in Figure 9(b), in the BIMCompNet dataset when the number of masonry layers was 30, the joint staggering compliance rate of the AIGSSA algorithm was 97.11%, and was much higher than the other three comparison algorithms. In summary, the AIGSSA algorithm dynamically enhanced the global search capability of layout schemes by constructing a multi-layer collaborative optimization architecture, demonstrating significant advantages in joint staggering compliance rate, thereby verifying the excellent performance of this algorithm in building infill wall masonry layout optimization.



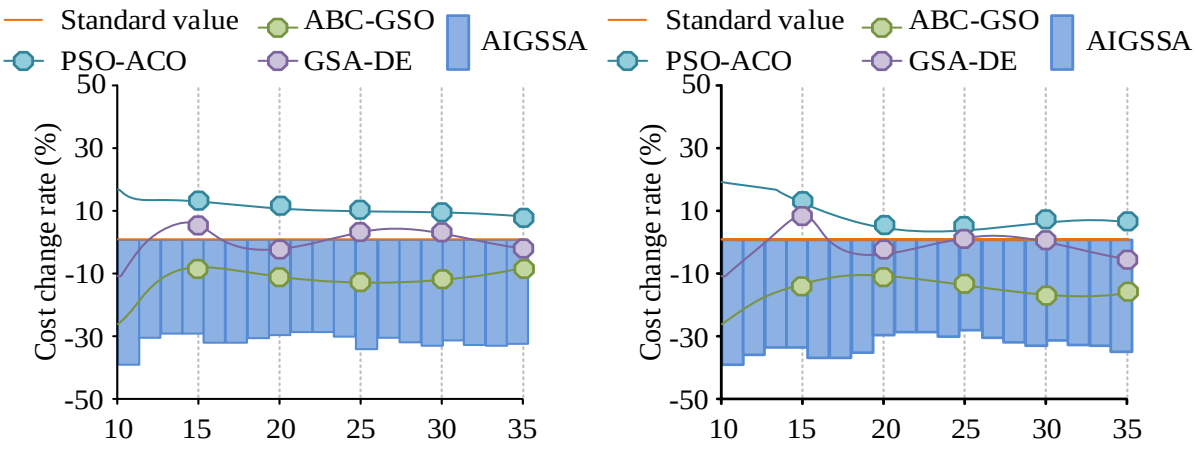
(a) The staggering compliance rate during layout optimization in the SLABIM dataset

(b) The staggering compliance rate during layout optimization in the BIMCompNet dataset

Fig. 9. Comparison of test results for joint staggering compliance rate

3.2 Verification of The Practical Layout Optimization Effect of AIGSSA

After verifying the optimization performance of the AIGSSA algorithm, the study proceeded to verify the engineering practicality of this algorithm. This study selected straight long walls with regular structural forms, straight axes, and no corners or abrupt cross-sections, and irregular walls including curved walls, polygonal corner walls, and obliquely cut walls as experimental scenarios. This study conducted comparative experiments with the three comparison algorithms PSO-ACO, ABC-GSO, and GSA-DE in the above two scenarios. To demonstrate the cost change advantage of the AIGSSA algorithm after layout optimization, this study tested the cost change rate of the four algorithms in the above two scenarios. The test results are shown in Figure 10.



(a) Regular long straight wall

(b) Alien wall

Fig. 10. Comparison of cost change rates in different scenarios

In Figure 10(a), in the regular straight long wall scenario when the number of masonries was 35, the cost change rate of the AIGSSA algorithm was -32.07%, which indicated that the AIGSSA algorithm reduced costs by 32.07% in practical application. The cost change rates of the other three comparison algorithms PSO-ACO, ABC-GSO, and GSA-DE were -9.68%, -3.56%, and 9.62%, respectively. In the irregular wall scenario in Figure 10(b), the cost change rate of the AIGSSA algorithm was -33.87%, which was also much better than the other algorithms. In summary, the AIGSSA algorithm could effectively reduce masonry layout costs, and the reduction magnitude was significantly better than other comparison algorithms, proving the practicality and efficiency of the AIGSSA algorithm in actual engineering. Next, to verify the capability of the layout optimization scheme obtained by the AIGSSA algorithm for long-term wall use, this study tested the masonry

cracking probability after simulated use for 3 years, 4 years, 5 years, and 6 years for the four algorithms. The study utilized finite element simulation to apply temperature cycling loads and humidity change loads to the wall model, simulating the thermal-humid coupling effect in the actual usage environment. At four time points of 3 years, 4 years, 5 years, and 6 years, the proportion of the wall elements with stress exceeding the tensile strength of the masonry was statistically calculated for each algorithm optimization scheme, which was used as the estimated value of the cracking probability. The test results are shown in Figure 11.

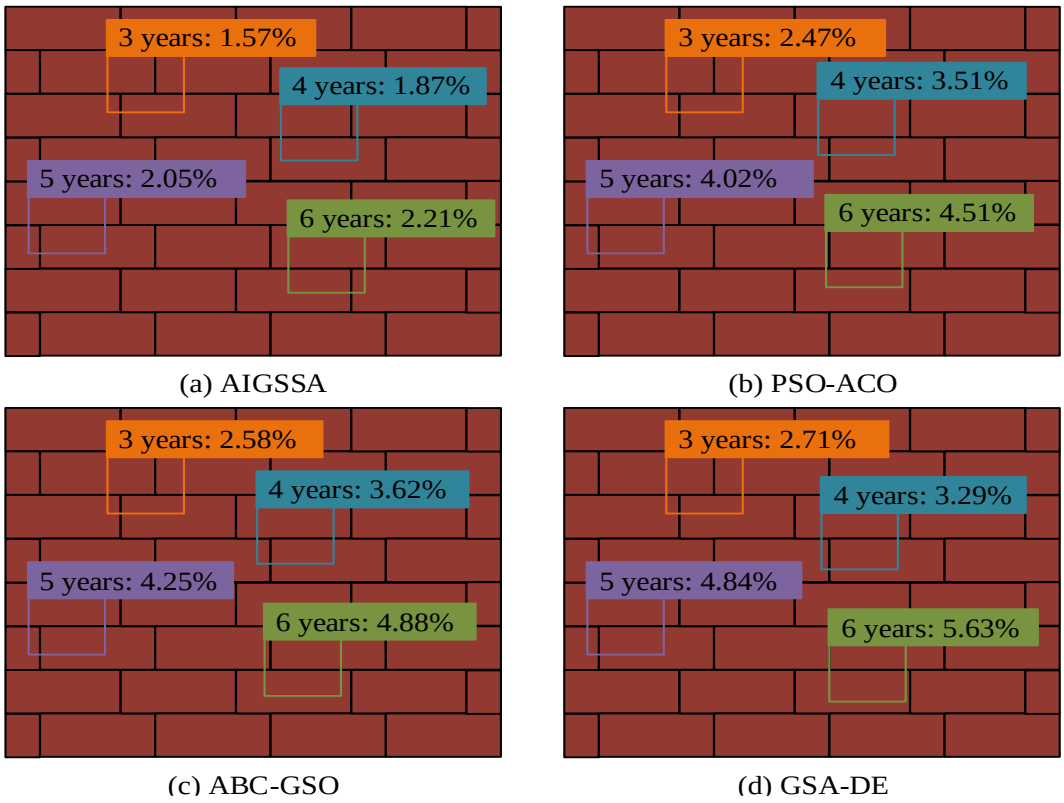


Fig. 11. Comparison of cracking probabilities in masonry of different ages

As shown in Figure 11(a), for the optimization scheme obtained by the AIGSSA algorithm, the masonry cracking probabilities after simulated use for 3 years, 4 years, 5 years, and 6 years were 1.57%, 1.87%, 2.05%, and 2.21%, respectively. In Figures 11(b), (c), and (d), the cracking probabilities after simulated use for 6 years for the optimization schemes obtained by the other three comparison algorithms PSO-ACO, ABC-GSO, and GSA-DE were 4.51%, 4.88%, and 5.63%, respectively. In summary, the AIGSSA algorithm demonstrated significant advantages in the cracking probability during simulated long-term wall use, verifying the effectiveness and engineering practicality of the AIGSSA algorithm in layout optimization. Finally, to verify the applicability of the AIGSSA algorithm in complex scenarios, this study tested the wall adaptation rate of the four algorithms when facing different wall types. Wall adaptability rate refers to the proportion of the arranged scheme optimized by the algorithm that can be applied in different types of walls, reflecting the algorithm's ability to adapt to complex engineering conditions. The calculation formula is: Wall adaptability rate = Number of applicable wall types / Total number of wall types × 100%. The test results are shown in Table 1.

According to the information in Table 1, in northern regions, the adaptation rates of the AIGSSA algorithm when facing four wall conditions of irregular walls, multi-opening walls, regular walls, and historic building restoration walls were 97.65%, 97.54%, 98.69%, and 96.62%, respectively. The wall adaptation rates of the other three comparison algorithms PSO-ACO, ABC-GSO, and GSA-DE were much lower than the AIGSSA algorithm. In southern regions, the adaptation rates of the AIGSSA algorithm when facing the same four wall conditions still maintained high adaptation levels and were much higher than the other three comparison algorithms. In summary, the AIGSSA wall

masonry layout optimization algorithm demonstrated extremely high adaptation rates when facing different types of walls in different regions.

Table 1. Comparison of compatibility results for different wall types (%)

Region	Algorithms	Irregular walls	Multi-opening walls	Regular walls	Historic building restoration walls
Northern region	AIGSSA	97.65±0.63*&#	97.54±0.58*&#	98.69±0.70*&#	96.62±0.55*&#
	PSO-ACO	92.23±0.81	91.64±0.80	89.65±0.84	88.27±0.79
	ABC-GSO	90.14±0.79	90.56±0.89	87.55±0.77	88.32±0.78
	GSA-DE	89.25±0.84	89.69±0.88	86.48±0.81	87.28±0.80
Southern region	AIGSSA	97.79±0.58*&#	97.46±0.60*&#	98.76±0.63*&#	96.75±0.65*&#
	PSO-ACO	92.36±0.60	93.52±0.61	90.12±0.59	88.34±0.62
	ABC-GSO	90.17±0.77	89.45±0.73	88.45±0.75	88.21±0.76
	GSA-DE	89.51±0.90	88.53±0.82	87.61±0.83	86.89±0.89

*Note: In Table 1, * indicates that the results of AIGSSA are significantly different from those of PSO-ACO ($p < 0.05$); & indicates that the results of AIGSSA are significantly different from those of ABC-GSO ($p < 0.05$); # indicates that the results of AIGSSA are significantly different from those of GSA-DE ($p < 0.05$).*

The research methods include GA, SA, SAS, AWOA, and ILS optimization algorithms. To evaluate the marginal performance improvement brought by each optimization layer, ablation experiments were designed. The comparison indicators include wall adaptability rate, arrangement efficiency, material utilization rate, and algorithm convergence speed. The experimental settings were consistent with the previous ones. Ablation comparisons were conducted for the five algorithms (GA, SA, SAS, AWOA, ILS). The ablation experiment results are shown in Table 2.

Table 2. Results of ablation experiments on the marginal performance improvement of each optimization layer

Ablation group	Wall fit rate (%)	Arrangement efficiency (%)	Material utilization rate (%)	Convergence speed (number of iterations)
GA	88.12	85.34	82.67	150
SA	90.45	87.21	84.33	130
SAS	91.78	90.56	87.89	110
AWOA	92.00	90.78	88.72	108
ILS	92.11	91.04	89.64	100
Complete model	95.77	96.42	93.15	80

As shown in Table 2, in the ablation experiments, as the optimization layers were gradually added, all performance indicators showed a steady improvement trend. From the GA model to the complete model, the wall adaptation rate increased from 88.12% to 95.77%, the arrangement efficiency increased from 85.34% to 96.42%, the material utilization rate increased from 82.67% to 93.15%, and the convergence speed was reduced from 150 iterations to 80 iterations. Among them, the leap from SAS to AWOA was the most significant, with the wall adaptation rate increasing by 0.22 percentage points, the arrangement efficiency increasing by 0.22 percentage points, the material utilization rate increasing by 0.83 percentage points, and the convergence speed accelerating by 2 iterations.

To evaluate the computational efficiency and scalability of the research method SIGSSA, the study conducted a comparative experiment by comparing it with modern reinforcement learning methods. The comparison methods included three deep reinforcement learning algorithms: DQN, PPO, and SAC. The comparison indicators were the average number of convergence iterations, the time taken for a single decision, and the final wall adaptation rate. The four algorithms were applied

to four different-sized wall arrangement scenarios in locations A, B, C, and D, and each scenario was run 20 times and averaged. The experimental results are shown in Table 3.

Table 3. Performance Comparison of Different Algorithms in Four Scenarios

Project		Average convergence iteration count	Single decision time (ms)	Final wall adaptation rate (%)
Scene A	DQN	120	45	92.34
	PPO	98	45	93.78
	SAC	105	38	93.12
	SIGSA	80	25	95.77
Scene B	DQN	121	45	92.11
	PPO	96	46	93.58
	SAC	104	40	93.36
	SIGSA	81	24	95.80
Scene C	DQN	119	48	92.00
	PPO	97	46	93.33
	SAC	106	39	93.24
	SIGSA	80	23	95.69
Scene D	DQN	122	49	92.06
	PPO	100	45	93.22
	SAC	102	40	93.36
	SIGSA	78	22	95.78

As shown in Table 2, SIGSSA demonstrated the best performance in all four scenarios. The average number of convergence iterations was 79.75, which was 33.9%, 18.6%, and 23.3% lower than DQN, PPO, and SAC respectively; the average time per single decision was 23.5 ms, which was 51.0%, 48.4%, and 40.5% lower than DQN, PPO, and SAC respectively; the final wall adaptation rate was 95.76%, which was 3.7%, 2.3%, and 2.5% higher than DQN, PPO, and SAC respectively. These results indicate that SIGSSA has significant advantages in terms of computational efficiency and wall adaptation quality. Moreover, it maintains stable performance in different scenarios, verifying its good generalization ability and robustness.

To evaluate the performance of SAS compared to the traditional A* algorithm and to prove the rationality of choosing SAS in the research, a set of comparative experiments was designed. In the same wall layout scenario, the path planning was conducted using both SAS and A* algorithms, and the average path length, average planning time, and path smoothness were recorded. The experimental results showed that SAS shortened the average path length by approximately 12% compared to A*, reduced the average planning time by approximately 35%, and improved the path smoothness by approximately 18%. This indicates that SAS outperforms the traditional A* algorithm in terms of path planning efficiency and path quality, further verifying its effectiveness and practicality in intelligent path guidance in complex wall environments.

4. Conclusions and Recommendations

Traditional building infill wall masonry layout optimization algorithms highly depended on the quality of initial schemes. When facing complex construction constraints, they had low layout efficiency, high material loss rates, and insufficient engineering adaptability. Therefore, this study proposed an intelligent masonry layout optimization algorithm for walls based on a multi-layer collaborative architecture. This algorithm generated diverse populations through GA, used SA for probabilistic jump optimization, introduced SAS to complete topology verification, and then combined AWOA with ILS to achieve deep exploration of the solution space. The experimental results showed that this algorithm performed excellently in the SLABIM and BIMCompNet dataset tests, with the highest standard block utilization rate reaching 98.76%, the lowest cutting loss rate at 2.79%, and the peak joint staggering compliance rate reaching 97.11%. In irregular wall and regular wall scenarios, this algorithm reduced costs by up to 23.87%, and the cracking probability

after simulated use for 6 years was 2.21%. In summary, the AIGSSA algorithm could accurately meet the multi-constraint requirements of masonry layout, effectively reduce construction complexity and material waste, and provide reliable technical support for the development of building industrialization. However, this algorithm still has shortcomings in collaborative layout across multiple building types. Future research will continue to optimize in engineering practice to improve the algorithm's adaptability to large-scale scenarios. To achieve scalability in industrial-scale masonry projects, the research will further explore the deep integration of this algorithm with the BIM system, develop an intelligent layout module that can provide real-time feedback, and introduce multi-objective optimization strategies to balance cost, schedule, and quality. At the same time, the research will attempt to combine reinforcement learning and transfer learning techniques to enable the algorithm to have the ability to transfer knowledge across projects, reducing the cost of re-training for new scenarios. Additionally, it will further explore the algorithm's layout capabilities in irregular walls, curved walls, and complex joints, and build a unified optimization framework for all types of building components.

Reference

[1] Sberna P, Demartino C, Vanzi I. Cost-effective topology optimization of masonry structure reinforcements by a linear static analysis-based GA framework. Bull Earthquake Eng. 2024;22(8):4143-67. https://doi.org/10.1007/s10518-024-01900-5

[2] Portioli FPA, Lourenço PB. Nonlinear static analysis of masonry structures with mortar joints and cracking units by optimization-based rigid block models. Earthquake Eng Struct Dynam. 2024;53(13):3963-82. https://doi.org/10.1002/eqe.4206

[3] Keshavarzfard R, Naderi A. A sustainable HazMat logistic network design considering scale of economy and route sensitivity and solving with Hybrid GA-SA. Int J Res Ind Eng. 2024;13(3):257-73.

[4] Atasi C, Kern J, Ramprasad R. Design of recyclable plastics with machine learning and genetic algorithm. J Chem Inf Model. 2024;64(24):9249-59. https://doi.org/10.1021/acs.jcim.4c01530

[5] Du Q, Qi X, Zou PXW. Bi-objective optimization framework for prefabricated construction service combination selection using genetic simulated annealing algorithm. Eng Constr Archit Manag. 2024;31(10):3921-45. https://doi.org/10.1108/ECAM-10-2022-1000

[6] Gangadevi E, Rani RS, Dhanaraj RK. Spot-out fruit fly algorithm with simulated annealing optimized SVM for detecting tomato plant diseases. Neural Comput Appl. 2024;36(8):4349-75. https://doi.org/10.1007/s00521-023-09295-1

[7] Kosanoglu F, Atmis M, Turan HH. A deep reinforcement learning assisted simulated annealing algorithm for a maintenance planning problem. Ann Oper Res. 2024;339(1):79-110. https://doi.org/10.1007/s10479-022-04612-8

[8] Pashaei E, Pashaei E. Hybrid binary COOT algorithm with simulated annealing for feature selection in high-dimensional microarray data. Neural Comput Appl. 2023;35(1):353-74. https://doi.org/10.1007/s00521-022-07780-7

[9] Lim KCW, Wong LP, Chin JF. Simulated-annealing-based hyper-heuristic for flexible job-shop scheduling. Eng Optim. 2023;55(10):1635-51. https://doi.org/10.1080/0305215X.2022.2106477

[10] Zanazzo E, Ceschia S, Dovier A. Solving the medical student scheduling problem using simulated annealing. J Sched. 2025;28(2):233-46. https://doi.org/10.1007/s10951-024-00806-z

[11] Valentino J, Gilbert M, Gueguin M. Limit analysis of masonry walls using discontinuity layout optimization and homogenization. Int J Numer Method Eng. 2023;124(2):358-81. https://doi.org/10.1002/nme.7124

[12] Son PVH, Soulisa FV. A hybrid ant lion optimizer (alo) algorithm for construction site layout optimization. J Softw Civ Eng. 2023;7(4):50-71.

[13] Tan Y, Yu D, Yang S. Precast concrete component yard layout optimization considering safety and efficiency. J Asian Archit Build Eng. 2025;24(4):2800-16. https://doi.org/10.1080/13467581.2024.2373837

[14] Shang D, Zou G. Optimizing Architectural Layouts and Spatial Configurations in Polar Environments Using Image Segmentation and Semantic Mapping Techniques. Int J Mach Learn Comput. 2025;42(4):1977. https://doi.org/10.18280/ts.420412

[15] Zheng Z, Li Y, Torres J. An integrated method of automated layout design and optimization for modular construction. Eng Constr Archit Manag. 2024;31(3):1016-36. https://doi.org/10.1108/ECAM-04-2022-0329

[16] Groumpos PP. A Critical Historic Overview of Artificial Intelligence: Issues, Challenges, Opportunities, and Threats. Artif Intell Appl. 2023;1(4):197-213. https://doi.org/10.47852/bonviewAIA3202689

[17] Aydoğan E, Cetek C. Aircraft route optimization with simulated annealing for a mixed airspace composed of free and fixed route structures. Aircraft Eng Aerospace Technol. 2023;95(4):637-48. https://doi.org/10.1108/AEAT-11-2021-0343
[18] Quankun M, Yanfei Z, Jinliang G. Traversal path planning of agricultural robot based on memory simulated annealing and A* algorithm. J South China Agric Univ. 2023;41(4):127-32.
[19] Lin Q, Ma H. SACHA: Soft actor-critic with heuristic-based attention for partially observable multi-agent path finding. IEEE Robot Autom Lett. 2023;8(8):5100-7. https://doi.org/10.1109/LRA.2023.3292004
[20] Zhang J, Li H, Parizi MK. HWMWOA: A Hybrid WMA-WOA algorithm with adaptive cauchy mutation for global optimization and data classification. Int J Inf Technol Decis Mak. 2023;22(4):1195-252. https://doi.org/10.1142/S0219622022500675