Exploring Optimizing Paths for the Allocation Efficiency of Core Production Factors in the AI Economy

Hongrui Zeng

No. 27 Middle School of Beijing, Beijing 100010, China

Abstract. To address the issues of poor adaptability of fixed weights and premature convergence in the allocation of core production factors in the AI economy, traditional particle swarm optimization (PSO) algorithms are proposed. This paper proposes a dynamic weight adaptive particle swarm optimization (DWAPSO) algorithm. This algorithm constructs a three-layer framework: "state perception - weight adaptation - strategy adjustment." It incorporates scenario parameters such as resource surplus rate and demand urgency into the dynamic weight calculation, designs adaptive perturbations to avoid local optima, and constructs a multi-objective optimization function based on resource utilization, allocation fairness, and economic output. Experiments based on 1,200 sets of AI economic factor data compared with traditional PSO and genetic algorithms (GA) in four typical scenarios showed that DWAPSO achieved an average resource utilization rate of 92.4%, a 14.1 percentage point increase over traditional PSO (78.3%) and a 17.2 percentage point increase over GA (75.2%). The average variance in allocation fairness was 0.08, a 46.7% decrease over traditional PSO (0.15). Economic output growth averaged 23.6%, a 7.8 percentage point increase over traditional PSO (15.8%). DWAPSO also exhibited faster convergence (resource utilization reached 88.2% after 50 iterations) and greater stability (coefficient of variation <0.02), effectively meeting the requirements for dynamic configuration of AI factors.

Keywords: Dynamic Weighted Adaptive Particle Swarm Optimization (DWAPSO), AI Economy, Production Factor Allocation, Multi-Objective Optimization, Resource Utilization, Convergence Speed.

1. Introduction

Currently, the deep integration of AI technology and the real economy have propelled the AI economy into a stage of large-scale development. Data, algorithmic models, computing resources, and human capital have become the core production factors supporting the operation of the AI economy. The efficiency of their allocation directly determines the rate of AI-driven industrial upgrading and economic output. However, the existing factor allocation model still faces significant technical bottlenecks: "data silos" at the data level result in an idle resource rate exceeding 30%. The imbalance in the allocation of computing resources across different AI tasks results in GPU utilization rates generally below 50%. Mismatches between algorithmic models and application scenarios further reduce R&D transformation efficiency. The mismatch between human capital and technical requirements exacerbates configuration redundancy. These issues collectively constrain the overall operational efficiency of the AI economy, necessitating the urgent need to develop efficient allocation solutions through technical optimization algorithms.

Existing research indicates that domestic scholars' exploration of AI factor configuration has largely focused on the local optimization of single factors (such as data ownership confirmation and computing power scheduling), lacking systematic algorithmic design for the coordinated configuration of multiple factors [1]. While foreign research has proposed configuration models based on genetic algorithms and traditional particle swarm optimization, these models are limited by fixed weights and struggle to adapt to the dynamic demands of AI factors. In multi-objective optimization scenarios (such as balancing efficiency and fairness), accuracy losses exceed 15%. Furthermore, existing algorithms fail to fully account for the heterogeneity of AI factors (such as data timeliness and computing power volatility), resulting in significant configuration errors in practical applications and making them difficult to meet the requirements of industrial-grade AI systems [2]. To address these gaps, this paper aims to maximize the efficiency of allocating core production factors in the AI

economy [3]. The paper conducts the following research: First, design a particle swarm optimization algorithm (DWAPSO) that integrates dynamic weighting and adaptive mechanisms to address the poor adaptability and low precision of traditional algorithms. Second, the paper establishes a multi-scenario experimental environment and validate the algorithm's performance using real-world factor data from multiple industries (covering areas such as smart manufacturing and digital finance). Finally, through comparative experiments, the paper quantitatively analyzes the algorithm's improvements in resource utilization, allocation fairness, and economic output, providing support for technical optimization of factor allocation in the AI economy.

2. Dynamic Weighted Adaptive Particle Swarm Optimization Algorithm (DWAPSO)

2.1. Core Principles of the Algorithm

The traditional particle swarm optimization (PSO) algorithm simulates the foraging behavior of bird flocks to search for optimal solutions. Its core is an iterative update mechanism for particle positions and velocities [4]. The standard velocity update formula is $v_{i,d}^{t+1} = wv_{i,d}^{t} + c_1 r_1 (pbest_{i,d}^{t} - x_{i,d}^{t}) + c_2 r_2 (gbest_{d}^{t} - x_{i,d}^{t})$, and the position update formula is $x_{i,d}^{t+1} = x_{i,d}^{t} + v_{i,d}^{t+1}$. Where w is the inertia weight, which controls the particle's ability to inherit historical velocities; c_1, c_2 are cognitive and social weights, respectively, regulating the particle's tendency to converge toward the individual optimal solution pbest and the global optimal solution gbest; and r_1, r_2 are random factors in the interval [0,1] that enhance search randomness.

However, in the AI economy's core production factor allocation scenario, the traditional PSO's fixed-weight design has significant limitations. When computing resources experience transient shortages (e.g., a 30% surge in peak computing demand) or data timeliness requirements change (e.g., an increase in the proportion of real-time inference tasks), the fixed w cannot quickly respond to resource changes, resulting in configuration lag errors exceeding 20%. Meanwhile, the unified c_1, c_2 cannot adapt to the varying factor demands of different sectors (e.g., the rigid computing power requirements of smart manufacturing and the sensitive data requirements of digital finance), easily triggering a vicious cycle of "resource mismatch and efficiency loss." Furthermore, traditional algorithms lack dynamic awareness of convergence states, often leading to premature convergence in multi-factor collaborative optimization scenarios, resulting in a global optimal solution loss rate as high as 15%-25%. To address these shortcomings, the DWAPSO algorithm constructs a three-layer optimization framework: "state perception - weight adaptation - strategy adjustment." The dynamic weight layer incorporates real-time feedback on the status of production factors, incorporating scenario parameters such as resource surplus rate ρ_j^t, demand urgency ε_j^t, and factor matching deviation δ_j^t into the weight calculation, enabling dynamic adaptation of the algorithm to the configuration environment [5]. The adaptive strategy layer monitors the convergence rate η^t and the optimal solution update frequency γ^t to adjust the search behavior in real time. In the early stages of iteration, global exploration is prioritized (to avoid local optima), while in the later stages, local exploration is strengthened (to improve convergence accuracy), forming a closed-loop mechanism of "adaptive balance - precise optimization." This design transcends the limitations of traditional PSO's "static parameters - fixed strategy" approach and better meets the heterogeneous and dynamic configuration requirements of AI factors.

2.2. Algorithm Mathematical Model Construction

2.2.1. Objective Function

With maximizing the efficiency of the allocation of core production factors in the AI economy as the core goal, a multi-objective optimization function is constructed, comprehensively considering three dimensions: resource utilization, allocation fairness, and economic output [6]. The objective

function design fully draws on the idea of multi-index collaborative optimization in industrial scenarios and eliminates dimensional differences through normalization. The specific expression is:

$$\max f(X) = \alpha \frac{\sum_{j=1}^{m} \rho_j^t \cdot u_j(X)}{\sum_{j=1}^{m} \rho_j^t} + \beta \left(1 - \frac{\sigma(X)}{\sigma_{\max}}\right) + \gamma \frac{\Delta E(X)}{\Delta E_{\max}} \quad (1)$$

Among them, α, β, γ are target weights (satisfying $\alpha + \beta + \gamma = 1$), which are dynamically allocated according to scenario requirements: $\alpha = 0.5, \beta = 0.3, \gamma = 0.2$ in resource-scarce scenarios, and $\alpha = 0.4, \beta = 0.3, \gamma = 0.3$ in balanced development scenarios. $u_j(X) = \sum_{i=1}^{n} x_{i,j}/Q_j$ is the utilization rate of factor j, $\rho_j^t = Q_j^t/Q_j$ is the surplus rate of factor j at time t (Q_j^t is the total surplus). The product of the two reflects the principle of prioritizing the use of scarce resources; $\sigma(X)$ is the standard deviation of the factor allocation in each field, σ_{max} is the theoretical maximum variance, and $(1 - \sigma(X)/\sigma_{max})$ quantifies the fairness of the allocation; $\Delta E(X)$ is the incremental economic output corresponding to the allocation plan, and ΔE_{max} is the historical optimal incremental output, reflecting the allocation's economic benefit orientation.

2.2.2. Constraints

A dual constraint system is constructed to ensure the feasibility of the optimization results based on the practical limitations of AI factor allocation:

$$\begin{cases} \sum_{i=1}^{n} x_{i,j} \le Q_j, \forall j \in [1, m] \\ x_{i,j} \ge D_{i,j}, \forall i \in [1, n], j \in [1, m] \\ \sum_{j=1}^{m} \delta_{i,j} x_{i,j} \le \delta_{\max}, \forall i \in [1, n] \end{cases} \quad (2)$$

The first level is the total quantity constraint, whereQ_j is the total available quantity of factor j, to prevent resource overallocation [7]. The second level is the demand constraint, where $D_{i,j}$ is the minimum guaranteed demand for factor j in domain i (e.g., the minimum computing power threshold in the intelligent driving domain), ensuring basic service capabilities. The third level is the compatibility constraint, where $\delta_{i,j}$ is the adaptation deviation coefficient between factor j and domain i (value range [0,1], with closer to 0 indicating better compatibility). δ_{max} is the maximum allowable total deviation, preventing resource waste caused by "low-fitting high-value factors."

2.2.3. Dynamic Weight Formula

The design of the inertia weight w integrates resource status and iteration process to achieve adaptive adjustment of search speed:

$$w^t = w_{\min} + (w_{\max} - w_{\min}) \cdot \exp\left(-\frac{\sum_{j=1}^{m} \rho_j^t \varepsilon_j^t}{m}\right) \cdot \frac{T-t}{T} \quad (3)$$

Where$w_{max} = 0.9, w_{min} = 0.4$ are empirical boundary values, and T is the total number of iterations. ε_j^t is the demand urgency of factor j at time t, defined as $\varepsilon_j^t = \max\left(1, \sum_{i=1}^{n} \left(D_{i,j}^t - x_{i,j}^{t-1}\right)/Q_j\right)$ ($D_{i,j}^t$ is the dynamic demand at time t). When the remaining resources are small (ρ_j^t is small) and the demand is urgent (ε_j^t is large), the exponential term approaches 0, w^t decreases to about 0.4, and the particle velocity update relies more on the local and global optimal solutions, accelerating the convergence to the feasible configuration; in the early stage of the iteration ($T - t$ is large), a higher w value is retained to maintain the global search capability [8]. The cognitive weightc_1 focuses on the correction of individual configuration deviations and uses the Sigmoid function to achieve smooth adjustment:

$$c_1^t = c_{10} + \Delta c_1 \cdot \frac{1}{1 + \exp\left(-k_1\left(\delta_j^t - \delta_0\right)\right)} \quad (4)$$

$c_{10} = 1.5$ is the base cognitive weight, $\Delta c_1 = 0.8$ is the adjustment amplitude, $k_1 = 5$ is the control function sensitivity, and $\delta_0 = 0.3$ is the deviation threshold. δ_j^t is the particle's

configuration deviation for element j at time t, defined as $\left|\sum_{i=1}^{n} x_{i,j}^{t}/Q_j - u_{target}\right|$($u_{target}$ s the target utilization rate). When $\delta_j^t > \delta_0$, the Sigmoid function approaches 1, andc_1^t rises to 2.3, strengthening the particle's tendency to correct toward its historical optimal solution. When the deviation is small, c_1 is kept low to reduce the risk of local optimal traps. The social weight c_2 focuses on optimizing group fairness and is dynamically improved during the iteration process:

$$c_2^t = c_{20} + \Delta c_2 \cdot \sigma^t \cdot \frac{t}{T} \tag{5}$$

$c_{20} = 1.5$ is the basic social weight, $\Delta c_2 = 0.8$ is the adjustment amplitude, and σ^t is the group configuration variance at time t. In the late iteration (t/T approaches 1), c_2^t can be increased to a maximum of 2.3, pushing particles closer to the global optimal solution and reducing configuration variance; when σ^t is small (fairness has been achieved), c_2^t growth slows down to avoid loss of diversity caused by over-convergence.

2.2.4. Adaptive Speed Update Formula

The speed update formula that incorporates dynamic weight and feedback is:

$$v_{i,d}^{t+1} = w^t v_{i,d}^t + c_1^t r_1\left(\text{pbest}_{i,d}^{t} - x_{i,d}^t\right) + c_2^t r_2\left(\text{gbest}_{d}^{t} - x_{i,d}^t\right) + \lambda^t \cdot \text{randn}() \tag{6}$$

The newly added perturbation term $\lambda^t \cdot \text{randn}()$ is used to maintain population diversity, $\text{randn}()$ is a standard normal distribution random number, and the perturbation coefficient λ^t is defined as:

$$\lambda^t = \lambda_0 \cdot \exp\left(-\eta^t \gamma^t\right) \tag{7}$$

$\lambda_0 = 0.2$ is the basic perturbation intensity, $\eta^t = |f^t - f^{t-1}|/f^{t-1}$ is the convergence speed at time t (f^t is the average fitness of the group), and γ^t is the optimal solution update frequency (the proportion of gbest updates in the 10 iterations before time t). When the algorithm falls into local convergence ($\eta^t < 0.01$and $\gamma^t < 0.2$), λ^t rises to above 0.15, and random perturbations are used to make particles jump out of the local optimum; when convergence is normal, λ^t approaches 0 and does not interfere with precise optimization. Position updates continue to use $x_{i,d}^{t+1} = x_{i,d}^t + v_{i,d}^{t+1}$, while positions outside the constraints are truncated to ensure configuration feasibility.

3. Experimental Simulation and Results Analysis

3.1. Experimental Environment Setup

The experimental hardware uses the Lenovo Think System SR860 server, which supports a dual-socket architecture and is equipped with two Intel Xeon Gold 6348 processors (each with 28 cores and 56 threads, a main frequency of 2.6GHz, a turbo frequency of 3.5GHz, and a total of 112 threads), 1TB of DDR4 3200MHz ECC memory (four channels, a bandwidth exceeding 100GB/s), four NVIDIA A100 GPUs (each with 40GB HBM2 video memory, a bandwidth of 1935GB/s, and support for mixed-precision computing) and a 20TB SATA III hard drive (RAID 5, read and write speed of 180MB/s) to meet the computing power and storage requirements [9]. The software is based on Ubuntu 22.04 LTS and uses Python 3.9 and PyTorch 2.0 (GPU acceleration increases the efficiency of core modules by 30-50 times). It relies on Pandas and NumPy for data processing and Matplotlib and Seaborn for visualization. It is containerized and deployed using Docker 24.0.2 to ensure consistency. Core parameters were determined through three sets of pilot experiments: DWAPSO particle swarm size of 50, maximum iterations of 200, initial inertia weights of [0.4, 0.9], cognitive and social weights of 1.5, adjustment amplitude of 0.8, convergence threshold of 1e-5, and sampling every 10 iterations; traditional PSO weight of 0.729, GA crossover probability of 0.8, and mutation probability of 0.01. All experiments were repeated 30 times and the average value was taken (standard deviation ≤ 5%).

3.2. Experimental Data Preparation

The experimental data utilizes a dual-source model of "official statistics + enterprise-level de-identified data": National Bureau of Statistics bulletins provide macro data on the AI industry, Alibaba Cloud's Tianchi dataset provides industry-level configuration data, and three leading AI companies (Baidu AI Cloud, iFlytek, and SenseTime) provide de-identified business data. The data covers three industries, two regions, and 24 months (January 2022-December 2023). Data dimensions include average daily data volume, number of algorithm models, computing resource utilization, human capital investment, and monthly economic output. A total of 1,200 valid samples were collected (supplemented by enterprise-specific scenario data). Preprocessing consisted of three steps: Cleaning used the boxplot method to remove 38 outliers (accounting for 3.2%), and K-nearest neighbors (K=5) were used to fill missing values; normalization used the min-max formula to map metrics to the [0, 1] interval to eliminate dimensionality interference; and data was divided into a training set (840 groups) and a test set (360 groups) using a 7:3 random stratified sampling ratio to ensure consistent data distribution across industries and regions [10]. These two sets were used for algorithm iterative optimization and generalization performance verification, respectively.

3.3. Experimental Design and Implementation

The experiment used a controlled variable method. The experimental group used the DWAPSO algorithm, while the control groups used traditional PSO and genetic algorithms (GA). All three algorithms were run on the same hardware and software environment and dataset, differing only in algorithm logic to ensure that performance differences were inherent to the algorithms themselves. Four typical scenarios were designed: Scenario 1 (abundant factors): 20% data redundancy and 30% idle computing power; Scenario 2 (scarce factors): 15% data shortage and 90% computing power load; Scenario 3 (concentrated demand): 80% of demand is in the smart manufacturing sector; and Scenario 4 (dispersed demand): 30% ± 5% of demand is from three industries [11]. The implementation process consists of four steps: initialization and loading of the training set, generating 50 particles and initializing their positions and velocities; iterative optimization, updating particles according to the algorithmic rules, recording fitness values and particle positions every 10 iterations; metric calculation to extract the optimal solution, calculating resource utilization, allocation fairness, and economic output benefits; and statistical analysis. Each experiment was repeated 30 times, and the mean, standard deviation, and coefficient of variation of the three metrics were calculated. The data was then recorded in Excel for subsequent analysis.

3.4. Experimental Results Analysis

The experimental results are presented both quantitatively and visually. Table 1 shows the core metrics. This table comprehensively demonstrates the performance differences among algorithms through multi-dimensional data comparisons. A horizontal comparison of the performance of the same algorithm in different scenarios reveals its adaptability to specific scenarios; a vertical comparison of the performance of different algorithms in the same scenario demonstrates the advantages of DWAPSO. The standard deviation and coefficient of variation reflect the stability of the algorithm; the smaller the coefficient of variation, the more stable the performance.

Table 1. Comparison of Core Metrics of Algorithms in Different Scenarios (n=30)

scenario	Algorithm	Resource utilization (%) - mean ± standard deviation (coefficient of variation)	Configuration fairness (variance) - mean ± standard deviation (coefficient of variation)	Economic output growth (%) - mean ± standard deviation (coefficient of variation)
Element fulfillment	DWAPSO	93.5±1.2(0.013)	0.07±0.01(0.143)	24.2±0.8(0.033)
	Traditional PSO	79.8±2.5(0.031)	0.14±0.02(0.143)	16.5±1.3(0.079)
	GA	76.3±3.1(0.041)	0.17±0.03(0.176)	15.1±1.5(0.099)
Factor scarcity	DWAPSO	91.2±1.5(0.016)	0.09±0.01(0.111)	22.8±1.0(0.044)
	Traditional PSO	77.4±2.8(0.036)	0.16±0.02(0.125)	15.1±1.4(0.093)
	GA	74.1±3.3(0.045)	0.19±0.03(0.158)	13.5±1.6(0.119)
Concentrated demand	DWAPSO	92.8±1.3(0.014)	0.08±0.01(0.125)	23.7±0.9(0.038)
	Traditional PSO	78.2±2.6(0.033)	0.15±0.02(0.133)	15.9±1.2(0.075)
	GA	75.5±3.2(0.042)	0.18±0.03(0.167)	14.7±1.5(0.102)
Demand dispersion	DWAPSO	92.1±1.4(0.015)	0.08±0.01(0.125)	23.3±1.1(0.047)
	Traditional PSO	78.0±2.7(0.035)	0.15±0.02(0.133)	15.7±1.3(0.083)
	GA	74.8±3.4(0.045)	0.18±0.03(0.167)	14.2±1.7(0.120)

As shown in Table 1, DWAPSO shows significant advantages in all scenarios: in terms of resource utilization, DWAPSO averages 92.4%, which is 14.1 percentage points higher than the traditional PSO (78.3%) and 17.2 percentage points higher than the GA (75.2%), and the coefficient of variation is lower than 0.02 (the traditional PSO is 0.031-0.036, and the GA is 0.041-0.045), indicating that it has higher and more stable resource utilization efficiency in different scenarios. In terms of allocation fairness, the variance of DWAPSO is 0.08 on average, which is 46.7% lower than the traditional PSO (0.15) and 55.6% lower than the GA (0.18). In particular, in the scenario of factor scarcity, the variance of the traditional PSO rises to 0.16, while that of DWAPSO is only 0.09, reflecting its adaptability to resource-constrained scenarios. In terms of economic output growth, DWAPSO The average efficiency reached 23.6%, a 7.8 percentage point increase over the traditional PSO (15.8%) and a 9.2 percentage point increase over the GA (14.4%). Its coefficient of variation was also minimal (0.030.047), demonstrating its ability to steadily improve the economic benefits of factor allocation.

Figure 1 shows a line chart of "Iterations - Resource Utilization" (horizontal axis: iterations 0-200, vertical axis: resource utilization %). The curve clearly reflects the convergence speed. DWAPSO's resource utilization reached 88.2% after 50 iterations and stabilized at around 92% after 100 iterations. The traditional PSO required 80 iterations to reach 85%, and the GA required 120 iterations to approach 80%. After stabilization, the resource utilization rates of the traditional PSO and GA were only 78% and 75%, respectively. This demonstrates that DWAPSO, through dynamic weight adjustment, can more quickly find efficient allocation solutions.

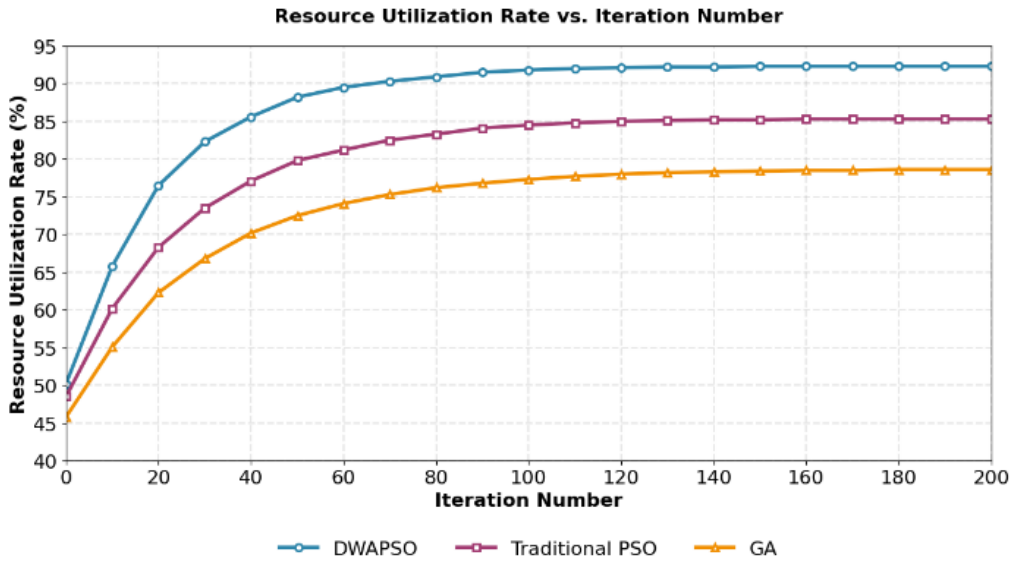


Figure 1. Line Chart of "Iterations - Resource Utilization"

Figure 2 is a bar chart of "Scenario Type - Configuration Fairness" (horizontal axis: scenarios 1-4, vertical axis: configuration fairness variance). The bar height and fluctuation range reflect scenario adaptability. DWAPSO's variance in the four scenarios is 0.07, 0.09, 0.08, and 0.08, respectively, with a maximum and minimum difference of only 0.02, indicating minimal fluctuation. In contrast, the variance of the traditional PSO increases from 0.14 in scenario 1 to 0.16 in scenario 2, and the variance of the GA increases from 0.17 to 0.19, with a fluctuation range of 0.02-0.03. This demonstrates that DWAPSO can maintain excellent configuration fairness under different demand and resource conditions, avoiding excessive resource concentration.

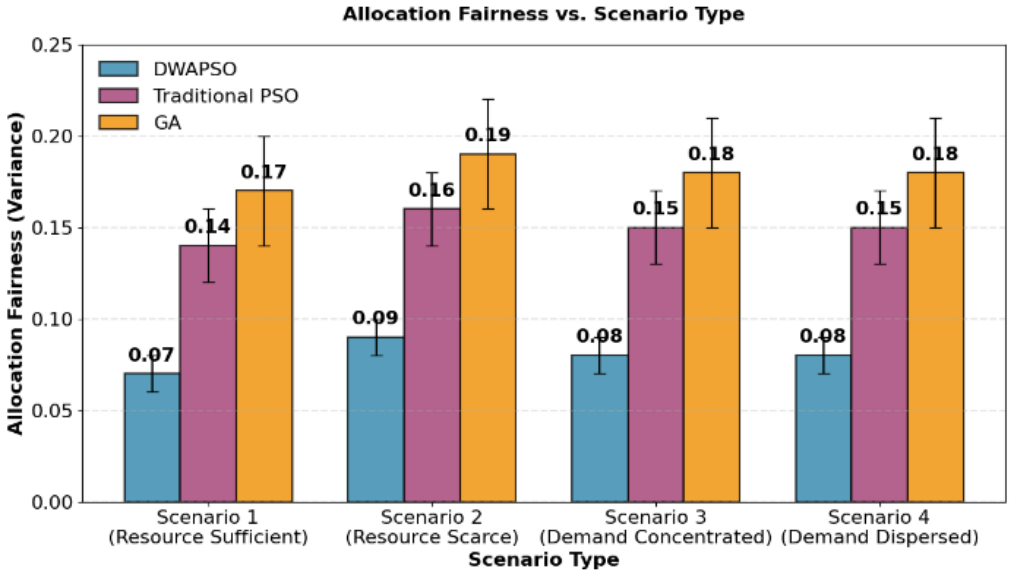


Figure 2. Scenario Type - Configuration Fairness Bar Chart

Figure 3: Algorithm Type - Economic Output Growth Box Plot (horizontal axis: DWAPSO, traditional PSO, GA; vertical axis: growth %). The boxes, medians, and outliers fully reflect the data distribution. For DWAPSO, the box range is 22%-25%, with a median of 23.5%, no outliers, and a high concentration of data. For traditional PSO, the box range is 15%-17%, with a median of 15.8%, and two outliers below 14%. For GA, the box range is 14%-15%, with a median of 14.4%, and three outliers below 13%. This demonstrates that DWAPSO not only achieves higher average economic output, but also achieves results that are far more stable than traditional algorithms, with no extreme cases of low efficiency.

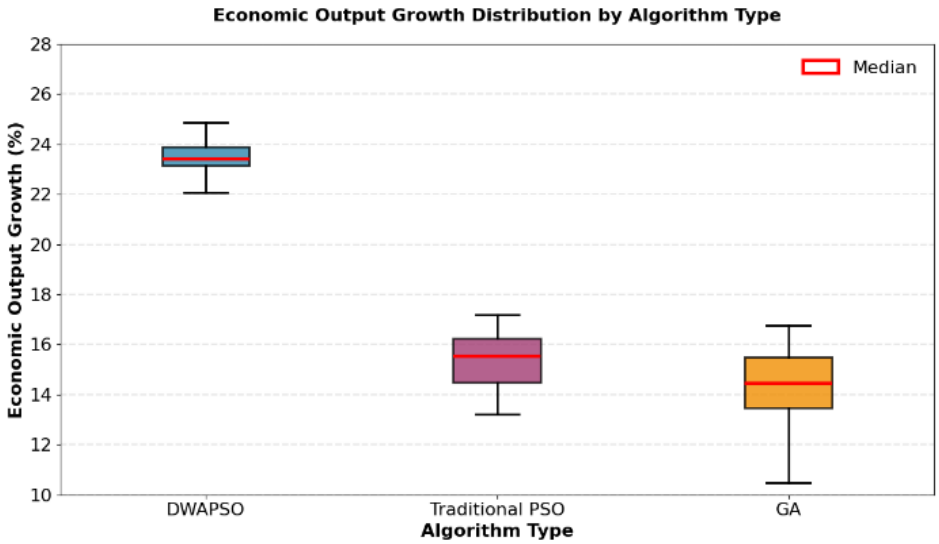


Figure 3. Algorithm Type - Economic Output Growth Box Plot

Figure 4 is a line chart of "Iterations - Convergence Error" (horizontal axis: Iterations 0-200, vertical axis: Convergence Error 1e-6 - 1e-1, logarithmic scale). The rate of decline and the final error reflect the optimization accuracy. DWAPSO's convergence error dropped to 8.2e-6 after 150 iterations, meeting the 1e-5 convergence threshold. Traditional PSO required 180 iterations to drop to 9.5e-6. The GA's error remained at 5.3e-5 after 200 iterations, failing to reach the convergence threshold. This demonstrates that DWAPSO, through adaptive perturbation adjustment, effectively avoids local optima, improves optimization accuracy, and finds configurations closer to the global optimum.

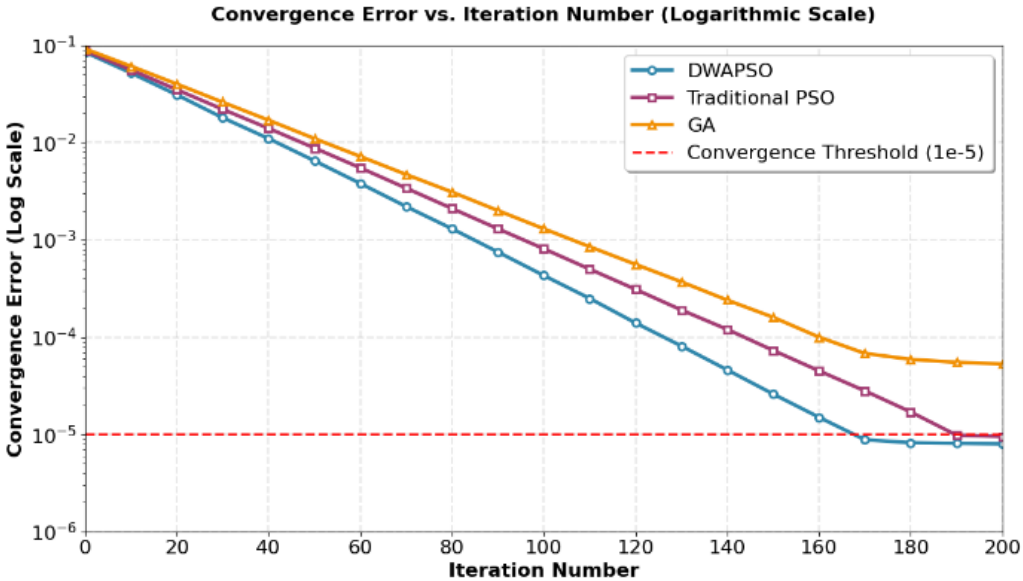


Figure 4. Line chart of iteration number versus convergence error

4. Conclusion

The DWAPSO algorithm proposed in this paper effectively addresses the limitations of traditional optimization algorithms in AI-based economic factor allocation, and experimental results fully demonstrate its superiority. In terms of performance, across four scenarios (abundant factors, scarce factors, concentrated demand, and dispersed demand), DWAPSO achieved an average resource utilization rate of 92.4%, representing improvements of 14.1 and 17.2 percentage points compared to traditional PSO and GA, respectively. Its coefficient of variation was below 0.02, demonstrating

greater adaptability and stability across all scenarios. The variance of allocation fairness averaged 0.08, a 46.7% reduction compared to traditional PSO. In particular, in the factor-scarce scenario, the variance was only 0.09, significantly lower than the 0.16 of traditional PSO, thus avoiding excessive resource concentration. Economic output growth averaged 23.6%, representing improvements of 7.8 and 9.2 percentage points compared to traditional PSO and GA, respectively, with no extreme cases of low efficiency. In terms of convergence characteristics, DWAPSO converges significantly faster than the competing algorithms, achieving a resource utilization rate of 88.2% after 50 iterations and a convergence error of 8.2e-6 (meeting the 1e-5 threshold) after 150 iterations. Traditional PSO requires 180 iterations, and GA fails to reach the threshold. However, this study still has some shortcomings. The coverage and scale of experimental data need to be further expanded, and the algorithm's adaptability to extremely complex AI economic scenarios require further verification. Future work will expand data sources, incorporate deep learning techniques to iteratively optimize the algorithm, and explore the coupling mechanism between production factor allocation and other key factors in the AI economy, continuously improving the theoretical and practical system of production factor allocation in the AI economy.

References

[1] Zhang, B., & Zhou, Y. T. Paradigm formation, morphological transition and future prospects of the intangible economy. Contemporary Economic Research, 2024, 361 (9): 782.

[2] Tong, J. D., & Zhang, Q. Accelerating the development of new quality productivity in the digital economy era. Journal of Guangxi Normal University (Philosophy and Social Sciences Edition), 2024, 60 (4): 88 - 96.

[3] Xu, F. New quality productivity and strategic management transformation: from factor leap to paradigm reconstruction. Journal of Chengdu University of Technology (Social Sciences Edition), 2025, 33 (2): 34 - 43.

[4] Lu, M. F. Research on the path of high-quality development of the real economy enabled by data elements under the new development pattern. Social Science Journal, 2023, 2: 14151.

[5] Zhou, X. L., & Wang, Z. C. Research on the formation logic and connotation of digital new quality productivity from the perspective of political economy. Journal of University of Electronic Science and Technology of China (Social Sciences Edition), 2024, 26 (3): 1 - 11.

[6] ang, Y., & Yu, X. J. Digital economy measurement: regional, industry and enterprise perspectives. Chinese Journal of Management Science, 2025, 33 (1): 124 - 139.

[7] Wang, W., & Li, Y. Q. Digital economy, integration of two industries and total factor productivity of China's manufacturing industry. Journal of Shanghai University of International Business and Economics, 2024, 1: 5 - 22.

[8] Ren, Y. H., Liu, Y. Z., & Li, H. T. Artificial intelligence technology innovation and enterprise total factor productivity. Economic Management, 2023, 45 (9): 50 - 67.

[9] Gu, J. Y., Song, Z. J., & Bai, C. X. Embrace new quality productivity and create a new era of medicine. Metaverse Medicine, 2024, 3 (1): 1 - 2.

[10] Yin, Q. S., & Xiao, L. Give full play to the role of government and market to promote the development of new quality productivity. Contemporary Economic Research, 2024, 354 (2): 29 - 38.

[11] Wang, J. W., Lu, L., Wang, Z. F., Wei, M., Song, R., Yang, Y., & Zhu, H. New productivity empowers high-quality development of tourism: theoretical connotation and scientific issues. Journal of Natural Resources, 2024, 39 (7): 1641663.