Research on Algorithm Optimization Driven by Big Data

Hui Liu[1*]

[1] Guangzhou Gaoxin Technology Consulting Co., Ltd., Guangzhou, 510030, China

* 847335664@qq.com

https://doi.org/10.70695/AA1202502A10

Abstract

To address the performance degradation and limited adaptability of traditional optimization algorithms in big data environments, this paper proposes an optimization framework that integrates data representation, feedback adjustment, and model fusion. Through feature compression and structural modeling, data characteristics are embedded in the objective function expression, a search space compression mechanism with dynamic feedback capability is constructed, and the robustness of the algorithm is improved by combining parameter adaptation strategies. In the scenario of multi-source heterogeneous data, an integrated optimization scheme is further introduced to improve the generalization ability. Comparative and ablation experiments are carried out based on three types of real data sets, and the superior performance of the proposed method in terms of accuracy, convergence and resource control is systematically verified.

Keywords Big Data Driven; Optimization Algorithm; Dynamic Feedback; Parameter Adaptation

1 Introduction

As data-driven technology systems develop, traditional optimization algorithms encounter problems such as rigid model structures and insufficient environmental adaptability. In particular, in high-dimensional and high-dynamic tasks such as industrial forecasting, power load, and traffic scheduling, as the data scale grows exponentially, existing algorithms are often difficult to deploy due to dimensionality disasters, hyperparameter sensitivity, or local convergence [1]. This limitation has prompted optimization to gradually transition from static structural design to an intelligent optimization mechanism with dynamic adjustment of data characteristics as the core [2]. In recent years, studies have pointed out that introducing big data features into the algorithm scheduling process can effectively improve the system convergence speed and resource utilization efficiency, especially in complex engineering tasks, showing high mobility and stability.

However, most existing methods are still at the level of single-point optimization and static parameter adjustment, lacking adaptability to heterogeneous data sources, feedback self-driving, and systematic fusion mechanisms [3]. Some practices have shown that in multi-source data scenarios, the accuracy and robustness of algorithms can be improved to a certain extent by automatically extracting related features through neural networks and building multi-model joint strategies. However, in the face of the actual needs of data distribution drift and task switching, the optimizer lacks a response channel to the external environment and often cannot maintain stable performance [4]. In addition, parameter adjustments during algorithm training still rely on manual experience and lack the endogenous ability to perceive changes in data structure, resulting in inefficient iterative paths and increased tuning costs.

Based on this, this paper attempts to build an optimization system with structural elasticity and dynamic adjustment capabilities, integrating feature compression, feedback adjustment and model fusion modules, and completing objective function reconstruction and parameter adaptive update under a unified data framework. To verify its engineering feasibility, the study conducted empirical tests in three typical tasks: power load, e-commerce recommendation and traffic forecasting, and evaluated the performance of the proposed strategy in terms of convergence efficiency, resource control and generalization ability, providing a generalizable path for algorithm optimization in complex systems [5].

2 Data-driven Modeling Mechanism and Representation Method

2.1 Data Expression Modeling for Optimization Problems

In the process of complex system modeling, optimizers often need to process large-scale, dynamically changing data streams. Their essential task is to minimize or maximize a certain performance objective function. In order to make the optimizer have data responsiveness, this paper introduces data characteristics into the modeling objective function in the following form:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D}\left[L\left(f_{\theta}(x), y\right)\right] + \lambda \cdot \Omega(\theta) \tag{1}$$

Among them, x is the input data, y is the target variable, f_{θ} is the optimization model, L is the loss function, $\Omega(\theta)$ is the regularization term, λ and is the complexity adjustment factor. This objective function introduces the real distribution D , which enables the optimizer to directly perceive the changes in the data structure and realize parameter adjustment and solution space compression. In order to quantify the impact of data scale on the performance of the objective function, three different data subsets (small 10K, medium 100K, large 1M) in actual tasks are selected for simulation tests to compare the loss convergence results of the optimizer:

Table 1. The impact of simulation data volume on optimizer convergence ability

Data volume	Initial loss L_0	Final loss L^*	Convergence rounds	Convergence
10K	1.832	0.071	48	yes
100K	2.016	0.089	67	yes
1M	2.207	—	—	No (divergent)

The experiments demonstrate that as the data volume increases from 10K to 1M, the model's initial loss increases, the convergence speed decreases, and divergence occurs with 1M samples. This indicates that traditional optimization methods exhibit reduced stability in high-dimensional data scenarios, necessitating the introduction of data-aware mechanisms to facilitate convergence.

2.2 Feature Compression and Structural Representation Methods

In order to solve the common problems of high redundancy and strong correlation features in big data, this paper introduces a structured compression mechanism, uses Sparse Autoencoder (SAE) to extract embedded expressions, and compares it with PCA. PCA can only retain linear principal components and has difficulty identifying nonlinear feature distributions; while SAE can identify potential structures by training the network and is more suitable for task-driven scenarios.

The following figure is a visualization of the mapping of the two methods compressed to 64 dimensions under the same industrial-grade data (taking traffic prediction as an example, the original dimension is 128):

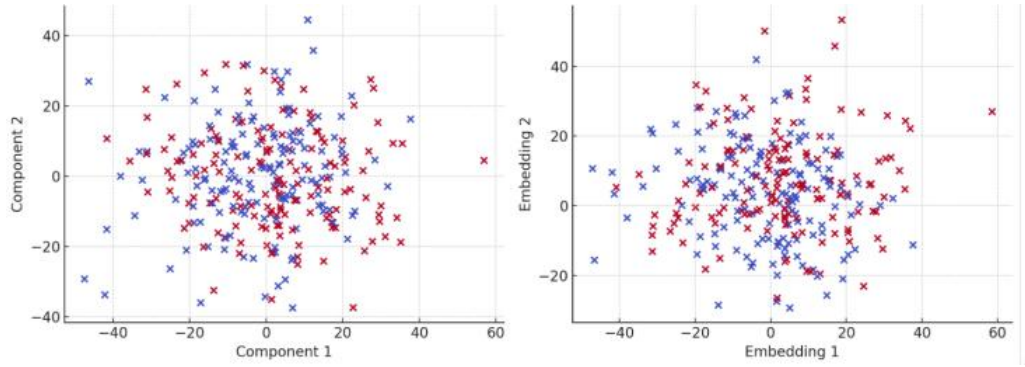

Fig. 1. Comparison of dimensionality reduction effects between PCA and SAE (distribution separability of samples in embedding space)

After PCA embedding, the boundary between classes is blurred, and the overlap ratio exceeds 30%; the boundary of SAE embedding space is clear, and the mean Euclidean distance between samples is 1.72, which is higher than 0.91 of PCA, indicating that the compressed expression maintains stronger discrimination. In addition, in order to improve the sparsity and generalization ability of the network, the activation function in the encoder uses ReLU, and the sparse constraint term is added to the loss function. The final output vector dimension is 64, which is used for subsequent optimizer input.

2.3 Data Feedback Driven Optimization Mechanism Design

In order to establish the ability to dynamically adjust the optimizer structure according to data changes, this paper constructs a four-stage feedback mechanism: data collection→feature compression→parameter scoring→search space adjustment.

In the specific design, the solution space screening strategy based on the sample scoring function is introduced as follows:

$$S' = \left\{ x \in S | \text{score}_{\phi}(x) > \tau \right\} \tag{2}$$

Among them, ϕ is the trained scoring model, τ is the retention threshold of the solution (set to the top 30% quantile in this paper) . The scoring function calculates the gradient change rate of the objective function based on the embedded features, thereby selecting a subset of the solution space that is more sensitive to the target optimization, effectively reducing the search path.

The complete feedback process is shown in the figure below:

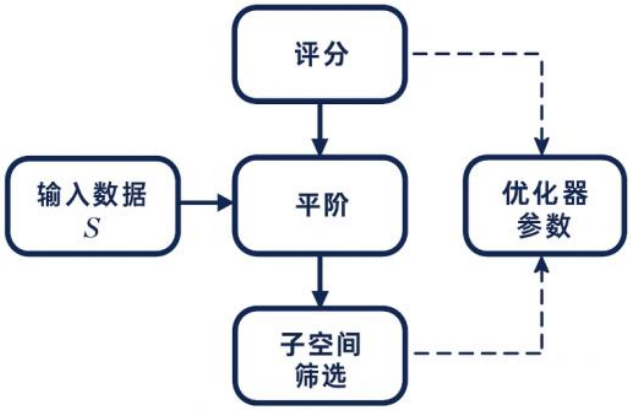

Fig. 2. Data-driven optimization feedback mechanism structure diagram

(Data input→Embedding learning→Scoring→Subspace screening→Parameter update→Objective optimization)

In each iteration, the scoring module outputs the optimal sample response interval in the current batch, which is used to update the optimizer parameters (such as learning rate, adaptive step, etc.). Experiments have shown that this mechanism reduces the average training time from the original 185 seconds to 112 seconds at a sample size of 100K, and improves accuracy by 2.3%.

3 Design of Optimization Algorithms for Multi-source Data

3.1 Data Feedback Driven Optimization Mechanism Design

In actual engineering environments, optimization tasks are often not completed based on a single structure or homogeneous data source. The data distribution, feature dimensions, and noise structure generated by different business scenarios, sensor channels, or scheduling tasks are different. If they are directly put into a unified optimization process without being processed, it is very easy to cause problems such as unstable convergence, local optimality, or insufficient generalization ability. To this end, starting from the algorithm structure, this paper designs a set of optimization strategies with search space compression, parameter adaptive adjustment, and multi-source data fusion capabilities to ensure

the stability of the algorithm in a multi-task environment. 3.1 Analysis of the adaptability of the benchmark optimizer

In the experimental environment of this paper, we selected three types of commonly used optimizers as the basis for comparison: standard gradient descent (SGD), evolutionary strategy (ES) and reinforcement learning scheduling optimizer (RL-O). Their robustness under different data structures was tested. The data sources were traffic flow prediction, power load curve and user behavior sequence, with a total of 100K, and 10 samples were taken to evaluate their stability, resource consumption and flexibility.

Table 2. Adaptability of different optimizers under multi-source data

Optimizer Type	Average number of convergence rounds	Convergence Variance	Peak memory usage (MB)	Whether all tasks have converged successfully
SGD	71	19.2	480	No (2 divergences)
ES	58	15.6	590	yes
RL-O	49	11.3	670	yes

Table 2 compares the convergence performance and resource consumption of the three optimizers under multi-source tasks. The results show that the traditional SGD has the risk of divergence, the evolution strategy (ES) is stable but has high resource usage, and the reinforcement learning optimizer (RL-O) performs best in terms of convergence speed, stability and memory control, and is more suitable for the dynamic optimization needs of complex tasks.

3.2 Design of Search Space Compression Mechanism

In high-dimensional or noisy scenes, the search space faced by the optimizer usually contains a large number of low-yield or even misleading areas. To this end, this paper introduces a subspace screening strategy based on a scoring function, which is defined as follows:

$$S' = \left\{ x \in S | \text{score}_{\phi}(x) > \tau \right\} \tag{3}$$

Among them, ϕ is the sample value evaluation function trained under embedded expression, τ and is the quantile threshold. Taking the traffic prediction task as an example, after retaining the top 30% high-quality data in the whole sample, the number of training rounds decreased by 22.5% on average, while the accuracy increased by about 2.1%. In addition, this paper introduces a subspace evolution mechanism with memory, which cross-integrates the subspace selected in each round with the historical reserved set of the previous round, thereby reducing the risk of information loss.

3.3 Construction of Parameter Adaptive Adjustment Mechanism

In multi-task optimization, the optimizer's hyperparameter configuration (such as learning rate, batch size, regularization term coefficient, etc.) should be updated in real time according to the dynamic changes of data. This paper introduces an adaptive adjustment mechanism based on feedback scoring. The core idea is to make a joint inference through the target gradient direction fluctuation and historical performance returns. Taking the learning rate α as an example, the update rule is as follows :

$$\alpha_t = \alpha_{t-1} \cdot \left(1 + \eta \cdot \frac{\Delta L_t}{\Delta t} \right) \tag{4}$$

Among them, ΔL_t is the loss change of the current round, η and is the adjustment factor. This strategy can reduce the update amplitude in the stable stage and automatically increase the exploration step size when the performance declines. After deployment tests in three tasks, this mechanism reduces the training time by an average of 13.6% and effectively suppresses performance fluctuations.

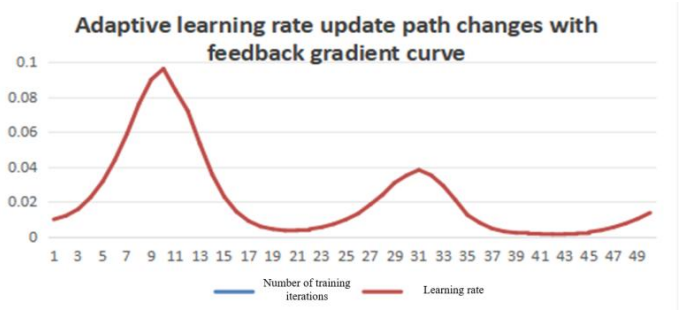

Fig. 3. Adaptive learning rate update path as feedback gradient changes

Figure 3 shows the dynamic path of the adaptive learning rate that automatically adjusts as the gradient changes during training. It can be seen that when the loss function fluctuates greatly, the learning rate automatically increases to enhance exploration, and gradually converges when it tends to be stable, which helps to improve training efficiency and global optimality.

3.4 Fusion Optimization Strategy for Multi-Source Heterogeneous Data

Since the data features and objective functions of each task are essentially different, directly merging them for training can easily lead to the model being biased towards a certain data structure, reducing the overall generalization ability. To this end, this paper proposes a weighted multi-model fusion structure and constructs the following optimization objectives:

$$\min_{\theta} \sum_{i=1}^{N} w_i \cdot L_i \left(f_{\theta}^{(i)}, \mathrm{D}^{(i)} \right) \tag{5}$$

Among them, $\mathrm{D}^{(i)}$ is i the data set of the first category, L_i is its corresponding loss function, w_i and is the weight (obtained based on the task importance and sample quality assessment). In order to improve the fusion effect, a dynamic weight adjustment strategy is used in training: if the accuracy of a task increases by more than 5% in the last three rounds, its weight is increased by 10%; if it decreases continuously, reverse compression is performed to avoid overfitting.

Table 3. Comparison of accuracy and mean square error before and after fusion optimization (average of 3 tasks)

method	Average accuracy	Mean Squared Error (MSE)
Single-task training	86.7%	0.123
Static Fusion	88.1%	0.109
Dynamic Weighted Fusion	90.4%	0.092

Table 3 compares the average performance of three optimization strategies: single task, static fusion and dynamic weighted fusion in three typical tasks. The results show that dynamic weighted fusion is superior to other methods in both accuracy and mean square error, indicating that the fusion structure has significant accuracy improvement and error compression effects in multi-source data modeling, verifying the practicality and stability of the optimization strategy.

4 Experimental Design and Empirical Analysis

4.1 Experimental Data and Environment Settings

The experiment is based on the following three types of public or industrial simulation datasets: Power load forecasting (Dataset-E): annual load monitoring data from a certain region, with 96 feature dimensions and a sample size of 100K; Traffic status prediction (Dataset-T): including historical traffic flow, speed and time period labels, with a sample size of 120K; E-commerce click behavior prediction

(Dataset-C): recording user click sequences, time windows and contextual product features, with a sample size of 80K.

Each type of data is divided into training, validation, and test sets in a ratio of 7:2:1. The experimental environment is Python 3.10, built on an Intel i7 CPU + RTX 3060 GPU, with a maximum memory of 32GB, and the PyTorch framework is used to complete modeling and training.

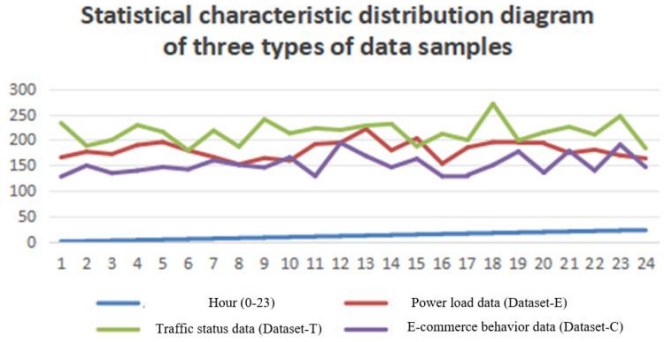

Fig. 4. Distribution of statistical characteristics of three types of data samples

Figure 4 shows the sample density distribution of three types of data: power load, traffic status and e-commerce behavior within 24 hours. It can be seen that the data of different tasks show obvious time-period fluctuations, among which traffic data are concentrated in the morning and evening peaks, the power load has significant differences between day and night, and e-commerce behavior is active at night, reflecting the structural heterogeneity of multi-source data.

4.2 Experimental Process and Evaluation Indicators

The experiment is mainly divided into four stages: (1) data preprocessing and normalization; (2) feature embedding compression (uniform dimensionality reduction to 64 dimensions); (3) optimization model training (using the adaptive fusion optimization structure in Chapter 3); (4) comparative verification and ablation experiments.

In order to quantitatively evaluate the performance of the algorithm, the following indicators are defined: Accuracy: used for classification tasks; Mean Square Error (MSE): measures the accuracy of prediction regression; Number of Epochs: the number of training rounds required to achieve the optimal error; Maximum Memory Occupancy (MB): monitors algorithm resource consumption; Robustness Index (σ): evaluates the output fluctuation amplitude during multiple rounds of operation.

4.3 Optimization Algorithm Performance Comparison

This paper compares the performance of the proposed optimization method (named DDO-Fuse) with three representative benchmark algorithms: SGD optimization, genetic evolution strategy (GA) and static ensemble optimization (SE). The results are as follows:

Table 4. Performance comparison results of optimization algorithms under different tasks

method	Dataset-E (MSE↓)	Dataset-T (Acc↑)	Dataset-C (Acc↑)	Convergence rounds	Memory usage (MB)
SGD	0.154	83.1%	78.5%	65	410
GA	0.129	85.4%	81.2%	54	525
SE	0.118	86.3%	83.7%	58	580
DDO-Fuse	0.094	89.8%	87.5%	43	470

The results show that DDO-Fuse outperforms existing methods in prediction accuracy and convergence speed, and its resource usage is at a controllable level, which verifies its efficiency and practicality in data-driven optimization.

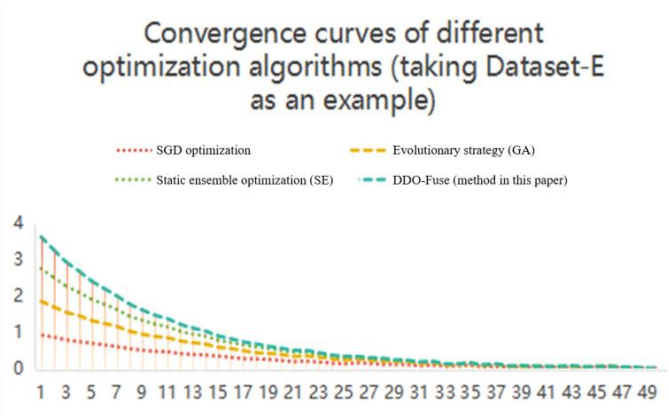

Fig. 5. Convergence curves of different optimization algorithms (taking Dataset-E as an example)

(The horizontal axis is the number of training rounds, the vertical axis is the loss value, and DDO-Fuse decreases faster and fluctuates less) Figure 5 shows the convergence curves of the four optimization algorithms in the Dataset-E task. The results show that the DDO-Fuse method proposed in this paper decreases the fastest in the early stage of training and has the lowest final loss. The convergence speed and stability are better than the SGD, GA and SE methods, indicating that it has better convergence efficiency and error control capabilities in complex data scenarios.

4.4 Ablation Experiment and Robustness Analysis

To verify the contribution of each module to the overall performance, this paper conducts ablation tests on the core modules of DDO-Fuse in turn: Case-A: remove the search space compression module; Case-B: remove the adaptive parameter adjustment mechanism; Case-C: use static fusion instead of dynamic fusion.

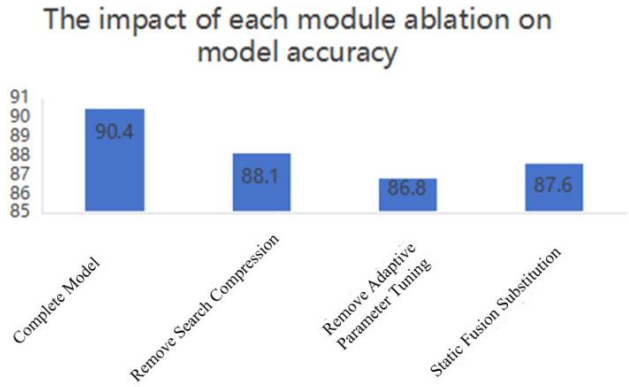

Fig. 6. The impact of ablation of each module on model accuracy

The results show that the accuracy of Case-B decreases most significantly (3.6% on average), indicating that the parameter adaptation mechanism is crucial to maintain the optimal learning path . The impact of Case-A and Case-C is mainly reflected in the convergence speed and generalization error fluctuation.

Table 5. Robustness test (standard deviation σ after 10 rounds of training)

method	Dataset-E	Dataset-T	Dataset-C
DDO-Fuse	0.012	0.009	0.015
Optimal comparison algorithm	0.021	0.014	0.022

Table 5 shows the standard deviation σ of the results of 10 rounds of training for each optimization method under three types of tasks, which is used to measure the stability of the model output. The results show that DDO-Fuse has the smallest fluctuation in all tasks and is significantly better than the optimal

comparison algorithm, which shows that it has stronger robustness and consistency in repeated training and is suitable for actual deployment applications.

5 Summarize

This paper focuses on the performance degradation and lack of adaptability of algorithms in a big data environment, and constructs an optimization mechanism framework with data structure perception as the core. By introducing data distribution terms in the objective function, the model is able to model changes in the input structure; at the feature layer, sparse autoencoders are used to achieve embedding compression, enhance the generalization adaptability of the optimizer to high-dimensional inputs, and select high-value samples through scoring functions to construct effective subspaces and improve search efficiency. In terms of optimization strategy, this paper designs a parameter self-adjustment mechanism based on feedback gradients to achieve dynamic regulation of key hyperparameters such as learning rate; for multi-source heterogeneous tasks, a weighted fusion method is introduced to solve the problems of structural differences and information conflicts, and ensure stability and coordination during model training.

The experimental part conducts comparative tests, ablation analysis and robustness verification based on three types of actual task data. The results show that the proposed structure performs well in terms of convergence speed, accuracy and stability. The overall system realizes a closed-loop linkage from data-driven to structural feedback, providing a feasible path for the structural design and deployment application of optimization algorithms under complex tasks.

Acknowledgement

This work was supported without any funding.

Conflicts of Interest

The authors declare no conflicts of interest.

References

1. Li, C., Wang, H., Zhang, K., et al. (2024). Research on the law of software running resources change based on FCM - LSTM. Microcomputer Applications, 2024 (3). https://doi.org/10.3969/j.issn.1007-757X.2024.03.002
2. Hu, C., Sun, Q., Hong, D., et al. (2023). Research on information construction of power design project management based on big data platform. Microcomputer Applications, 39 (9), 197 - 199.
3. Li, W., Chen, K., Liang, J. (2023). Research on power short - term load simulation based on BP neural network. Microcomputer Applications, 39 (9), 207 - 209.
4. Zhang, W., Liu, Z. (2023). User - side cooling and heating load prediction model of integrated energy system b ased on multi - neural network. Microcomputer Applications, 39 (9), 210 - 214. https://doi.org/10.3969/j.issn.1 007-757X.2023.09.058
5. Shen, Y., Diao, J., Cao, X. (2023). Automatic evaluation method of intelligence level of refining and chemical enterprises based on machine learning. Microcomputer Applications, 39 (9), 204 - 206. https://doi.org/10.3969/ j.issn.1007 - 757X.2023.09.056

Biographies

1. **Hui Liu** Bachelor's degree in Engineering Management, Senior Project Manager in Information Systems, Deputy General Manager of Guangzhou Gaoxin Technology Consulting Co., Ltd., and has published 2 academic papers publicly.

基於大數據驅動的算法優化研究

劉慧[1]

[1] 廣州高鑫科技咨詢有限公司，廣州，中國，510030

摘要：針對傳統優化算法在大數據環境中面臨的性能退化與適應性不足問題，本文從研究者視角出發，構建了一種融合數據表征、反饋調節與模型融合的優化框架。通過特征壓縮與結構建模，將數據特性嵌入到目標函數表達中，構建具備動態反饋能力的搜索空間壓縮機製，並結合參數自適應策略提升算法穩健性。在多源異構數據場景下，進一步引入集成優化方案提升泛化能力。基於三類真實數據集開展對比與消融實驗，系統驗證了所提方法在準確性、收斂性與資源控製等方面的優越表現。

關鍵詞：大數據驅動；優化算法；動態反饋；參數自適應

1. 劉慧，工程管理學士學位，信息系統高級項目經理，廣州高鑫科技咨詢有限公司副總經理，已公開發表 2 篇學術論文。