

Article

A Multi-Objective Optimization Method for Desensitization Resource Allocation of Civil Aviation Passenger Sensitive Data

Shuang Wang ¹, Huan Zhao ¹, Lei Ding ², Zhaojun Gu ¹, Jingxian Zhou ¹ and Zhiping Li ^{2,*}

¹ Information Security Evaluation Center, Civil Aviation University of China, Tianjin 300300, China; s-wang@cauc.edu.cn (S.W.); zhaohuan_cauc@163.com (H.Z.); zjgu@cauc.edu.cn (Z.G.); jxzhou@cauc.edu.cn (J.Z.)

² Department of Electrical Engineering, Tsinghua University, Beijing 100084, China; dloftjcu@163.com

* Correspondence: lizp@cauc.edu.cn

Abstract

To address the challenges of heterogeneous desensitization requirements, limited system resources, and inefficient resource allocation for structured sensitive data, this paper proposes a data security-oriented multi-objective optimization method for desensitization resource allocation of civil aviation passenger sensitive data. The proposed method integrates heterogeneous resource scheduling with multi-objective optimization to achieve adaptive resource allocation under resource-constrained environments. A clustering-based approach is employed to group desensitization tasks with similar computational characteristics, and a personalized sensitivity coefficient is introduced to characterize task complexity and guide adaptive resource allocation. Based on task-group characteristics, a multi-objective optimization framework integrating the Non-dominated Sorting Genetic Algorithm II (NSGA-II) is developed to jointly optimize desensitization execution time, computational resource consumption, and task-group matching cost. Experimental results show that the proposed method successfully optimizes 92.76% of desensitization tasks, while 31.59% achieve optimal solutions with fewer thread resources than the original configuration. Hyperparameter sensitivity analysis further identifies an effective parameter configuration that provides a favorable balance between optimization quality and computational efficiency. Comparative experiments with five representative multi-objective optimization algorithms further verify the superiority and robustness of the proposed method.

Keywords: civil aviation passenger sensitive data; personalized sensitivity coefficient; task grouping; k-means clustering; NSGA-II; multi-objective optimization



Academic Editor: Luis Javier Garcia Villalba

Received: 10 June 2026

Revised: 9 July 2026

Accepted: 10 July 2026

Published: 13 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Civil aviation passenger sensitive information desensitization technology involves methods to protect passengers' personal sensitive information [1] such as names, ID numbers, passport details, and contact information. These methods aim to reduce or eliminate the identifiable aspects of sensitive data, thus minimizing risks of leakage or misuse. Recent incidents of passenger data breaches have resulted in significant compensation costs for airlines. As the volume of passenger data grows and cybersecurity laws and regulations become stricter, the challenges and responsibilities related to processing this data are rapidly expanding. There is an increasing demand for efficient, reliable, and scalable desensitization solutions tailored to the specific needs of civil aviation. These systems are often deployed in

environments with limited resources, making confidentiality, power efficiency, performance, and sustainability essential factors for optimization. Genetic methods are employed for these optimization tasks. They work by iteratively improving potential solutions through processes like crossover, mutation, and selection. The NSGA-II algorithm and desensitization operators are introduced to generate new solutions by combining randomly selected individuals in the population based on their difference. Based on their performance, evaluated with NSGA-II, these new solutions replace older ones. The use of genetic algorithms is favored for their strong applicability and effectiveness in handling complex engineering problems.

Research on civil aviation passenger data protection and anonymization technology is constantly advancing. Zhang et al. [2] addressed the challenge of protecting sensitive passenger information in paperless and convenient travel context by proposing an attribute encryption method that enables user-specific access control for secure information sharing. Li [3] and others developed an adaptive desensitization scheme for structured datasets, integrating hierarchical analysis and clustering algorithms to categorize sensitive and non-sensitive attributes, using a segmentation preservation format encryption algorithm for desensitization. Zhang [4] explored a multistage data desensitization technique, allowing for different privacy-preserving algorithms to be applied on demand based on data type, thus improving dataset usability while protecting user privacy. Lv [5] focused on semantic-based medical text desensitization to improve desensitization efficiency and usability by optimizing the automatic desensitization process and improving the related algorithms. Liu [6] introduced an ε_k personalized data desensitization protection method to address the limitations of low availability value and incomplete desensitization of sensitive attributes through basic anonymization in the direct masking desensitization method. The dataset undergoes k-anonymization, and then the quasi-identifiers of the sensitive data in the equivalence group are ε_k -normalized. The desensitization protection of private data is achieved through employing the corresponding constraint method. This method meets the requirements for the individuality protection of sensitive data. It reduces the sensitivity of the information. It also enhances individual users' control over their data sensitivity. Yang et al. [7] proposed a dynamic desensitization algorithm for virtual power plants, utilizing machine learning approach for scene recognition and data privacy level classification. A novel similarity utility function and a Gaussian process-based differential privacy algorithm ensure tailored and efficient privacy protection.

Research on intelligent bionic algorithms and optimization methods continues to evolve, especially in areas like resource allocation within power grids and machinery. PAGA addresses the decoupling capacitor (decap) optimization issue in power distribution networks, and after selecting an initial solution from a prioritization point of view, a genetic algorithm is used to improve convergence, demonstrating strong robustness in solving both real-world and large-scale problems [8]. The hybrid genetic algorithm and particle swarm optimization Adaptive algorithm (HGAPSO) is designed to automatically determine the optimal topology to achieve a specific filter response by constructing branches and generating suitable element values for a reflection-free filter. HGAPSO demonstrates excellent selectivity [9]. Furthermore, the improved genetic planning algorithm effectively finds optimal solutions under multiple constraints such as priority, time, and resources, particularly in scenarios involving shared resource scheduling with limited time [10]. Li et al. [11] proposed a multi-objective method based on NSGA-II, which compresses the search space through an improved sparse initialization strategy, guides the selection of optimal solutions, dynamically shrinks the search space via mutation during population evolution, and employs a greedy repair mechanism to generate superior feature subsets. The method demonstrates strong performance across 15 publicly available high-dimensional datasets. Jiwapatria et al. [12] proposed an improved PMR-NSGA-II algorithm, which addresses the

optimal solution for energy distribution and parameter configuration in control systems by introducing dynamically generated reference points and a population-guided mechanism. The parallel bi-objective real-coded genetic algorithm is used to overcome the challenges of simulation optimization in multi-intelligent body fuzzy transportation systems. It identifies optimal solutions using known test cases and applies these to calculate near-optimal parameters for multi-intelligent systems. This model also supports maneuverability, aiding in turning and lane-changing tasks within these systems [13]. Meanwhile, Ayesha Maroof et al. [14] explored cutting-edge logistics optimization using a hybrid GA. This approach effectively addresses the vehicle routing problem with time windows, classified as an NP-hard problem, offering fresh solutions for improving logistics and supply chain operations. Zhang et al. [15] proposed a deep genetic algorithm for high-speed channel signal-integrity optimization, where a deep neural network was employed to predict signal-quality metrics and significantly accelerate the genetic optimization process. Vijai et al. [16] proposed a multi-objective hybrid feature selection technique that integrates information gain, random forest, and Relief-F. By combining the advantages of filter-based and wrapper-based feature extraction methods, the approach introduces feature score-guided crossover and mutation operators to improve convergence efficiency and enhance search directionality. Jiang et al. [17] proposed an NSGA-II-based MNNPS control parameter optimization strategy. They constructed a fitness function incorporating a nuclear reactor physical model and designed mutation operators based on system dynamic responses. Furthermore, both bi-objective and tri-objective optimization strategies for MNNPS control parameters were developed, and NSGA-II was applied to sequentially optimize the control system parameters. Gu et al. [18] integrated NSGA-II into a topological model to address the congestion scheduling problem in TSN-CAN gateways.

The processing and utilization of passenger sensitive data span the entire travel life-cycle. To ensure data security, desensitization techniques such as masking, suppression, encryption, and substitution are widely adopted to reduce data identifiability and mitigate risks of data leakage and misuse. However, existing desensitization systems typically rely on predefined desensitization strategies and static resource allocation schemes, making them increasingly inadequate for modern distributed multi-server environments characterized by node heterogeneity and limited computational resources.

Specifically, three critical challenges arise. First, node heterogeneity, including differences in hardware configurations, software environments, and computational capabilities, leads to significant performance variability under unified desensitization strategies. Second, resource constraints limit the ability of individual nodes to efficiently handle high-complexity desensitization tasks. Third, diversified desensitization requirements across different stakeholders make it difficult for a single strategy to satisfy various application scenarios. These issues collectively result in low efficiency, high resource consumption, and poor alignment between desensitization performance and business requirements. Therefore, developing a multi-objective optimization method that jointly optimizes desensitization execution time, computational resource consumption, and task-group matching cost under resource-constrained distributed environments has become a pressing challenge.

The main contributions of this work are summarized as follows:

- (1) A heterogeneous resource scheduling and dynamic allocation framework is proposed under resource-constrained environments. By incorporating business processing intent as a constraint, the proposed framework enables adaptive resource allocation for heterogeneous desensitization tasks.
- (2) A clustering-based task-grouping mechanism is designed to group desensitization tasks with similar computational characteristics, thereby providing task-group information for subsequent resource allocation optimization.

- (3) A multi-objective optimization model is constructed to jointly minimize desensitization execution time, computational resource consumption, and task-group matching cost. Based on this model, an NSGA-II-based optimization framework is developed to obtain a diverse set of Pareto-optimal solutions under multiple resource constraints.

The remainder of this paper is organized as follows. Section 2 presents the business-intent-oriented constraint strategy and analyzes the relationships among resource features. Section 3 introduces the task-grouping method and the personalized sensitivity coefficient. Section 4 formulates the proposed multi-objective optimization model based on NSGA-II. Section 5 describes the experimental setup, and Section 6 presents and analyzes the experimental results. Finally, Section 7 concludes the paper and outlines future research directions.

2. A Business-Intent-Oriented Constraint Strategy and Dynamic Resource Allocation Model

Business processing intent reflects the heterogeneous desensitization requirements of different data processors for civil aviation passenger sensitive data. Owing to the diversity of desensitization tasks, the demand for computational resources varies considerably across application scenarios. Consequently, a static resource allocation strategy cannot satisfy the requirements of heterogeneous desensitization tasks. Therefore, desensitization resources should be allocated dynamically and adaptively according to task characteristics and business requirements. To address this issue, a business-intent-oriented constraint strategy and dynamic resource allocation model is proposed.

During the desensitization process, each desensitization task is executed by applying a combination of desensitization rules, which together constitute a desensitization method. For civil aviation passenger sensitive data, different desensitization rules are selected according to the types and quantities of sensitive information contained in the task. Let k denote the number of desensitization rules applied to a task. The value of k is jointly determined by the data characteristics and the business processing intent. Consequently, different combinations of desensitization rules produce different desensitization method, resulting in different computational resource requirements.

The mapping relationship between the desensitization rules R and the desensitization method M is given in Equation (1).

$$M(x) = F(R_1, R_2, \dots, R_i), 1 \leq i \leq n \quad (1)$$

where R_i denotes the i -th desensitization rule, and $F(\cdot)$ represents the mapping function that generates a desensitization method from a valid combination of desensitization rules.

As illustrated in Figure 1, civil aviation passenger sensitive information includes names, identification numbers, bank card numbers, itinerary information, home addresses, facial images, fingerprint data, and frequent-flyer information. Different desensitization rules are selected according to the type and quantity of sensitive information. These rules are then combined to construct a desensitization method that satisfies the corresponding business requirements, including service scope, business volume, and processing objectives.

For example, an airport may process passenger information such as names, identification numbers, facial recognition data, and itinerary information. Based on the characteristics and volume of the sensitive data, an appropriate set of desensitization rules is selected to construct the corresponding desensitization algorithm. Different business processing intents therefore result in different method configurations and resource requirements.

In summary, business processing intent determines the configuration of desensitization rules, which subsequently defines the desensitization method and its corresponding compu-

tational resource requirements. This relationship provides the foundation for the resource allocation and multi-objective optimization model developed in the following sections.

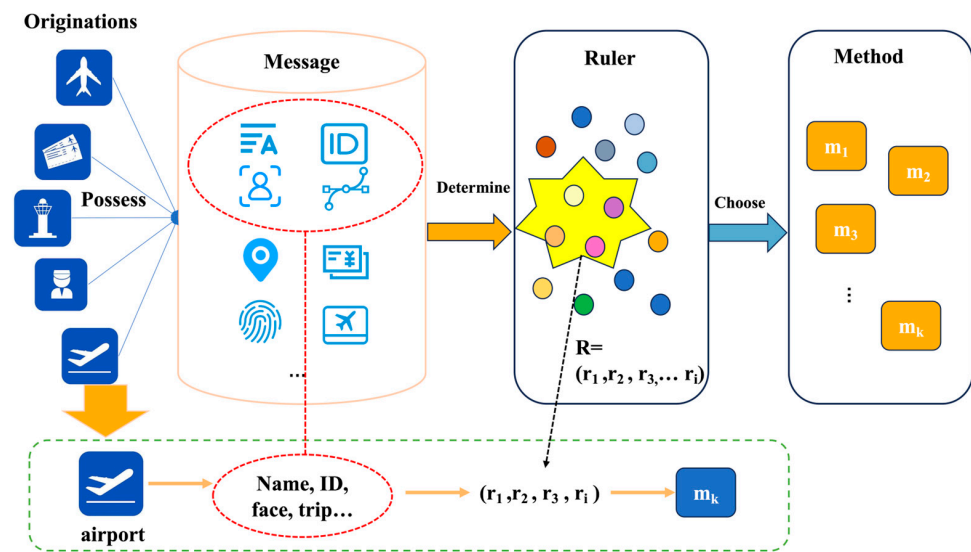


Figure 1. Relationship between business processing intent, desensitization rules, and desensitization method for civil aviation passenger sensitive data.

2.1. Feature Definition and Representation

Feature normalization is first performed on the resource consumption data of desensitization tasks, followed by feature extraction from the dataset. The extracted features include: file name, which serves as the unique identifier of each desensitization task for traceability and result mapping; file size (Byte), representing the overall data volume of the task; total number of rows in the file, indicating the total number of records and reflecting task granularity; number of rows per slice, which determines the granularity of each partition for parallel processing; number of slices, referring to the total number of partitions into which a file is divided and serving as a key decision variable for parallelization; desensitization time (milliseconds), defined as the wall-clock execution time and used as the primary optimization objective; and number of threads, representing the level of parallel computing resources allocated and another key decision variable.

It should be noted that the dataset does not explicitly provide the mapping between each task and its corresponding desensitization methods and rules. Table 1 summarizes the representation of each feature.

Table 1. Comparison table of field representation.

Name	Representation
File name	$Name(i)$
File size (Byte)	S_i
Number of rows in the file	R_i
Number of rows per slice	r_i
Number of slices	P_i
Desensitization time (ms)	T_i
Number of threads	Th_i

Where i denotes the file number, $Name(i)$, S_i , and R_i are inherent properties of each file and do not change during the process. r_i should satisfy the following relationship (2):

$$r_i \cdot P_i \approx R_i \quad (2)$$

where r_i is the number of rows in each file partition for task i , P_i represents the total number of file partitions, and R_i denotes the total number of rows in the corresponding file.

Therefore, the critical decision variables in the original dataset are P_i and Th_i . The files are numbered and the characteristic relationships between the metrics are compared.

2.2. Feature Correlation Analysis of Resource Data

To investigate the relationships among resource-related features and identify the key factors affecting desensitization execution time, correlation analyses are conducted for four pairs of variables: file size versus execution time, total number of rows versus execution time, total number of rows versus file size, and number of slices versus execution time. The analysis aims to reveal the characteristics of resource consumption and provide the basis for subsequent multi-objective optimization.

- (1) Relationship between file size and desensitization execution time. As shown in Figure 2, the desensitization execution time generally decreases with the sample index, whereas file size fluctuates considerably. Several tasks with relatively small file sizes still exhibit long execution times. For example, around sample index 100, the file size is significantly smaller than that of neighboring samples, whereas the corresponding execution time remains relatively high. These observations indicate that file size alone cannot accurately characterize the computational cost of desensitization tasks. The observed relationships can be summarized as follows:
 - Larger files containing more sensitive information generally require longer desensitization execution time;
 - Smaller files containing highly sensitive information may also require relatively long execution time;
 - Larger files with relatively low sensitivity can be processed efficiently under appropriate resource allocation;
 - Smaller files with low sensitivity generally exhibit shorter execution time.
- (2) Relationship between the total number of rows and desensitization execution time. As illustrated in Figure 3, although the execution time generally decreases with the sample index, the total number of rows varies substantially among different tasks. Some tasks with relatively few rows still require comparatively long execution times. For example, the task around sample index 60 contains relatively few rows but exhibits an execution time comparable to neighboring tasks. Therefore, the total number of rows alone cannot fully explain the execution time. The observed relationships can be summarized as follows:
 - Tasks containing a large number of rows and highly sensitive information generally require longer execution time;
 - Tasks with fewer rows but higher data sensitivity may also require relatively long execution time;
 - Tasks containing many rows but relatively low sensitivity can be completed within shorter execution time;
 - Tasks with fewer rows and lower sensitivity generally require the shortest execution time.
- (3) Relationship between the total number of rows and file size. As shown in Figure 4, both variables exhibit noticeable fluctuations. Overall, the dataset follows the expected trend that larger files generally contain more rows. However, several exceptions are observed. For instance, around sample index 275, the file contains relatively few rows but is larger than neighboring files. These observations indicate that file size is jointly affected by both the number of records and the average amount of information contained in each record. The relationship can be summarized as follows:

- Files containing more records with relatively compact content generally have larger file sizes;
 - Files containing many records with a large proportion of blank or sparse fields may remain relatively small;
 - Small files generally contain fewer records or relatively simple data structures;
 - Files containing fewer records may still be relatively large when individual records include substantial information.
- (4) Relationship between the number of slices and desensitization execution time. As illustrated in Figure 5, increasing the number of slices does not necessarily reduce the desensitization execution time. In several cases, tasks with a larger number of slices still exhibit relatively long execution times. This phenomenon may be attributed to multiple factors, including larger file sizes, higher data sensitivity, increased scheduling overhead caused by excessive slicing, inappropriate resource allocation, and insufficient thread resources. Therefore, the number of slices alone cannot determine the execution performance of a desensitization task.

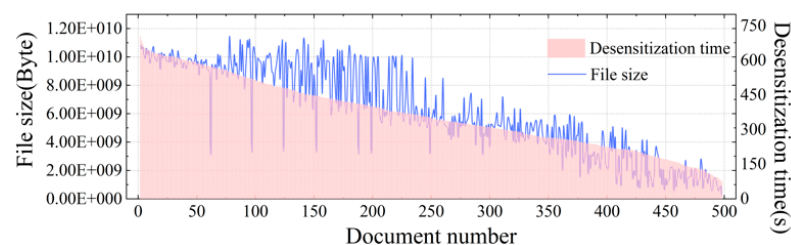


Figure 2. Relationship between file size and desensitization execution time.

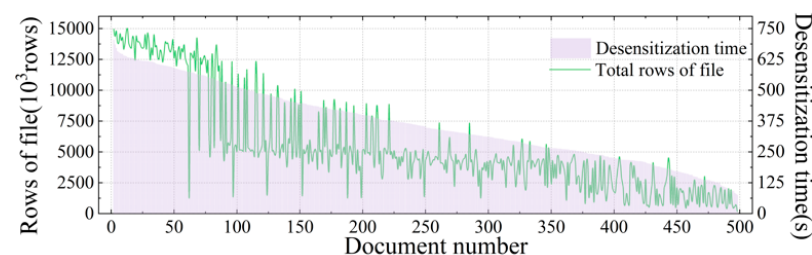


Figure 3. Relationship between the total number of rows and desensitization execution time.

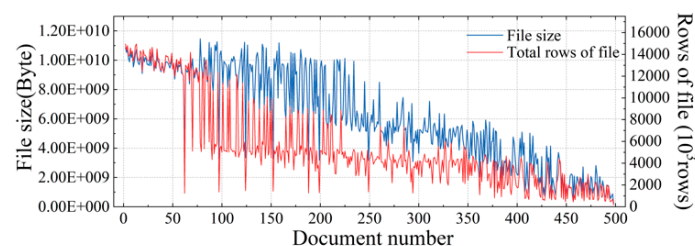


Figure 4. Relationship between the total number of rows and file size.

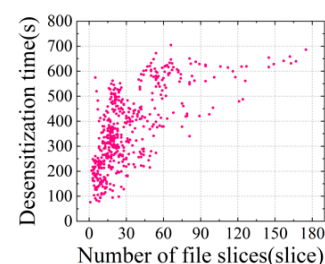


Figure 5. Relationship between the number of slices and desensitization execution time.

Based on the above feature correlation analysis, no obvious linear relationship is observed among the resource-related characteristics of the desensitization system. Instead, the execution time is jointly influenced by multiple factors, including file size, total number of rows, number of slices, and thread allocation. These observations indicate that optimizing a single variable is insufficient to achieve efficient desensitization. Therefore, the number of slices, thread allocation, and task-group assignment are selected as the primary decision variables in the subsequent multi-objective optimization model, while the desensitization execution time serves as the primary optimization objective.

3. Dynamic Mapping Relationship Between Desensitization Algorithms and Tasks Based on Clustering

Due to data security protection requirements, the operational data collected from the desensitization system do not explicitly record the desensitization method adopted for each task. Instead, only task-related resource features, such as file size, total number of rows, number of file partitions, and number of threads, are available. Under this black-box setting, it is therefore difficult to directly characterize the computational characteristics of different desensitization tasks. To address this issue, a clustering-based task-grouping strategy is proposed to identify tasks with similar computational characteristics.

Specifically, an unsupervised K-means clustering method is employed to group desensitization tasks according to their normalized resource feature vectors. Tasks with similar computational characteristics are assigned to the same task group, thereby providing task-group information for the subsequent personalized sensitivity analysis and multi-objective resource allocation.

The k-means clustering method [19] partitions a dataset into K clusters such that samples within the same cluster are as similar as possible, while samples belonging to different clusters are as dissimilar as possible. In this study, K-means is employed to discover the intrinsic grouping patterns of desensitization tasks according to their resource characteristics.

First, K initial cluster centroids are selected using the k-means++ initialization strategy. For each task, the Euclidean distance to every cluster centroid is calculated, and the task is assigned to the nearest centroid [20]. After all tasks are assigned, the cluster centroids are updated, and the intra-cluster dispersion is recalculated. This iterative process continues until the convergence criterion is satisfied. The overall clustering procedure is illustrated in Figure 6.

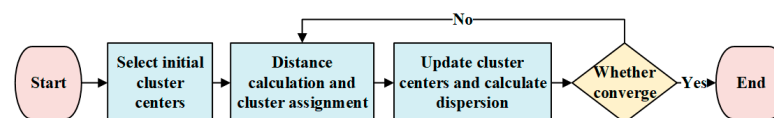


Figure 6. Unsupervised clustering process based on k-means.

Feature Extractions

The seven extracted features are summarized in Table 1. Because the operational logs do not contain explicit labels indicating the desensitization method adopted for each task, task grouping is performed using only the resource-related characteristics. Accordingly, each task is represented by a four-dimensional resource feature vector,

$$x_i = [S_i, R_i, P_i, Th_i]^T \quad (3)$$

where x_i represents the normalized resource feature vector of task i , comprising four attributes: file size S_i , total number of rows R_i , number of slices P_i , and number of threads Th_i . To eliminate the influence of different feature scales, all features are normalized prior to clustering.

Let C_k denote the centroid of the k -th task cluster, where $k = 1, 2, \dots, K$. For each normalized task feature vector x_i , the Euclidean distance to every cluster centroid is calculated. The distance between task i and cluster k is defined as

$$d_k^i = \|x_i - C_k\|_2 = \sqrt{\sum_{j=1}^m (x_{ij} - C_{kj})^2} \quad (4)$$

where m denotes the number of feature dimensions. The Euclidean distance is used to measure the similarity between a task and each cluster centroid. Each task is assigned to the cluster with the minimum Euclidean distance.

After the cluster assignment, the compactness of each cluster is evaluated using the intra-cluster dispersion. Since each task is represented by a four-dimensional normalized feature vector, the dispersion is computed as the average squared Euclidean distance between all samples within the cluster and the corresponding cluster centroid:

$$D_k = \frac{1}{|C_k|} \sum_{i \in C_k} \|x_i - C_k\|_2^2 \quad (5)$$

where C_k denotes the set of samples belonging to the k -th task cluster, and $|C_k|$ represents the number of samples in that cluster. A smaller value of D_k indicates higher cluster compactness and greater similarity among tasks within the same cluster.

The objective of K-means clustering is to minimize the total intra-cluster dispersion over all clusters, which is expressed as

$$J = \sum_{k=1}^K \sum_{i \in C_k} \|x_i - C_k\|_2^2 \quad (6)$$

where J represents the overall clustering objective. The clustering process iteratively updates the cluster centroids until the objective function converges.

Finally, each task is assigned to the nearest task cluster according to the minimum-distance criterion:

$$A_i = \arg \min_{k \in \{1, 2, \dots, K\}} d_{ik} \quad (7)$$

where A_i denotes the task group assigned to task i . It should be emphasized that the obtained clusters represent groups of tasks with similar computational characteristics rather than true desensitization methods. Since the dataset does not provide the true desensitization method labels, the clustering results are interpreted as task groups for subsequent resource allocation and optimization.

The number of task groups is set to $K = 13$ according to domain knowledge. Specifically, the operational desensitization system provides 13 predefined desensitization methods, each corresponding to a distinct processing strategy. Although the operational logs do not record which desensitization method is adopted for each task, the number of task groups is therefore fixed to 13 to reflect the underlying business configuration. K-means is subsequently employed to group tasks with similar computational characteristics into these task groups.

Figure 7 illustrates the convergence process of the K-means algorithm after fixing $K = 13$. As shown in Figure 7a, the within-cluster sum of squared errors (SSE) decreases rapidly during the initial iterations and gradually stabilizes, indicating that the clustering process converges to a stable solution. Figure 7b presents the distribution of tasks among the 13 task groups after convergence.

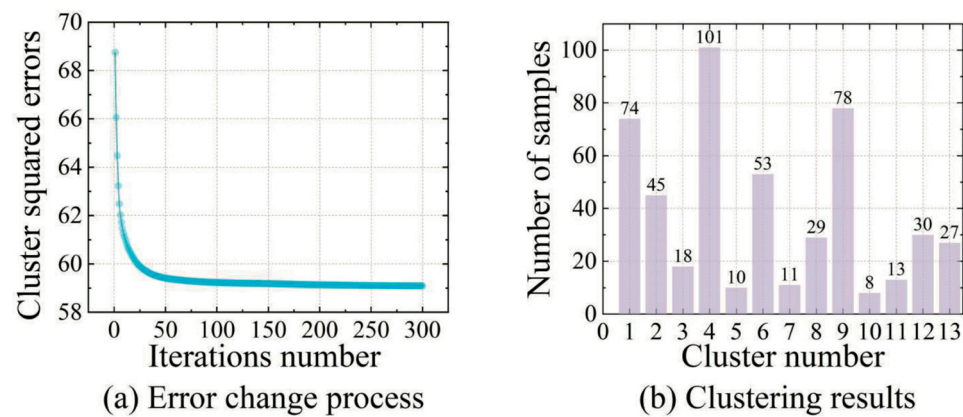


Figure 7. K-means clustering results for desensitization tasks.

4. Multi-Objective Optimization Method Based on the Integration of the Non-Dominated Sorting Genetic Algorithm II (NSGA-II)

The Non-dominated Sorting Genetic Algorithm II (NSGA-II) [21] is a fast elitist multi-objective optimization algorithm that preserves elite solutions during the evolutionary process. NSGA-II starts by initializing a population and evaluating the fitness of each individual. Non-dominated sorting is then performed to identify the Pareto fronts, while the crowding distance is calculated to maintain population diversity. Selection, crossover, and mutation operators are subsequently applied to generate offspring populations. The parent and offspring populations are merged and re-ranked until the predefined termination criterion is satisfied. Consequently, NSGA-II can effectively search for a well-distributed set of Pareto-optimal solutions.

In this study, NSGA-II is adopted to optimize resource allocation for desensitization tasks with heterogeneous computational characteristics under resource-constrained environments.

4.1. Personalized Sensitivity Coefficient

Different desensitization tasks exhibit heterogeneous computational characteristics due to variations in data scale and resource allocation requirements. To characterize the intrinsic computational complexity of each task, a personalized sensitivity coefficient is introduced. Unlike the original formulation, the proposed sensitivity coefficient is constructed solely from task-related features and resource allocation parameters, without using the observed execution time, thereby avoiding information leakage during execution time estimation. In this study, the sensitivity coefficient is defined as a task complexity indicator that characterizes the computational sensitivity of desensitization tasks to resource allocation.

Accordingly, the personalized sensitivity coefficient is defined as

$$\alpha_i = \frac{S_i R_i}{P_i Th_i} \quad (8)$$

where S_i , R_i , P_i and Th_i is denote the file size, total number of rows, number of slices and number of threads of task i . A larger value of α_i indicates that the task requires more computational resources to complete the desensitization process under the current resource allocation strategy.

Figure 8 illustrates the distribution of the personalized sensitivity coefficients. Tasks with larger file sizes and more rows generally exhibit higher sensitivity coefficients, whereas increasing the number of slices and threads reduces the sensitivity coefficient by improving the parallel processing capability. Therefore, the proposed sensitivity coefficient provides a quantitative measure of the intrinsic computational complexity of desensitization tasks.

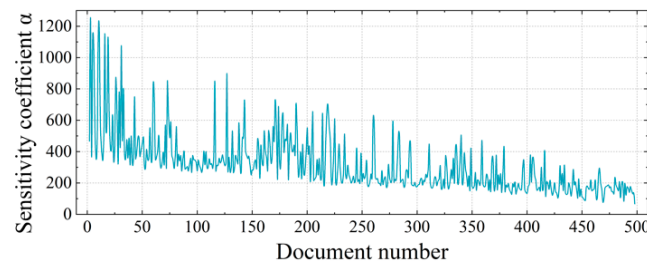


Figure 8. Sample sensitivity coefficient distribution.

To further measure the consistency between a task and its assigned task group, the task-group matching cost θ_i is introduced. It is calculated by the proportional deviation between the task feature vector and the corresponding cluster centroid, as formulated in Equation (9):

$$\theta_i = \sum_{j=1}^m \left(\frac{|x_{ij} - \mathcal{C}_{A_{ij}}|}{|\mathcal{C}_{ij}| + \varepsilon} \right)^2 \quad (9)$$

where x_{ij} denotes the j -th feature of task i , $\mathcal{C}_{A_{ij}}$ represents the j -th feature of the centroid of the assigned task group assigned to task i , and ε is a small positive constant to avoid division by zero. A smaller value of θ_i indicates a higher similarity between the task and its assigned task group.

Combining the personalized sensitivity coefficient and the task-group matching cost, while considering the scheduling overhead introduced by slice allocation and thread scheduling, the estimated desensitization execution time of task i is expressed as follows:

$$\hat{T}_i = (1 + \theta_i) \cdot \alpha_i + T_{sch}(i) \quad (10)$$

where $\hat{T}_i$ denotes the estimated desensitization execution time of task i , and $T_{sch}(i)$ represents the scheduling overhead.

The scheduling overhead is defined as

$$T_{sch} = \beta \cdot P_i + \gamma \cdot Th_i \quad (11)$$

where β and γ scheduling overhead coefficients of one slice and one thread, respectively.

4.2. Multi-Objective Optimization Problem

According to the problem description, an efficient resource allocation and task processing strategy should minimize desensitization execution time while controlling computational resource consumption. Therefore, the problem is formulated as a multi-objective optimization problem with three objectives: execution time, resource consumption, and task-group matching cost.

$$\min F = (F_1, F_2, F_3) \quad (12)$$

where $F_1 = \sum_{i=1}^N \hat{T}_i$, $F_2 = \sum_{i=1}^N (P_i + Th_i)$, $F_3 = \sum_{i=1}^N \theta_i$.

Here, F_1 represents the total estimated desensitization execution time, F_2 represents the total computational resource consumption, and F_3 represents the task-group matching cost.

The constraints of the multi-objective optimization problem are defined as follows:

$$s.t. \begin{cases} 1 \leq Th_i < 448 \\ 1 \leq A_i < K \\ \zeta_{min} \leq r_i \leq \zeta_{max} \\ r_i = \frac{R_i}{P_i} \end{cases} \quad (13)$$

where Th_i denotes the number of threads allocated to task i , A_i denotes the task group assigned to task i , and K is the total number of task groups. ζ_{min} and ζ_{max} denote the minimum and maximum thresholds of the number of rows in each slice, respectively. r_i represents the number of rows per slice, which is determined by the total number of rows R_i and the number of slices P_i .

4.3. Solution to the Multi-Objective Optimization Problem Based on the Integration of NSGA-II

The workflow of the NSGA-II-based multi-objective optimization method is illustrated in Figure 9.

- (1) First, the personalized sensitivity coefficient is calculated for each task according to Equation (8). Then, K-means clustering assigns each task to a corresponding task group based on its normalized resource feature vector.
- (2) After task initialization, a certain number of individuals are randomly selected to form the initialize parent population [22]. Each individual represents a candidate resource allocation strategy, including the number of slices, number of threads, and task-group assignment.
- (3) Each individual represents a candidate resource allocation strategy, including the number of slices, number of threads, and task-group assignment.
- (4) For each generation, the fitness of each individual is evaluated according to the three objective functions defined in Equation (12). Non-dominated sorting and crowding distance calculation are then performed. Based on elitist selection, crossover, and mutation operations, offspring populations are generated.

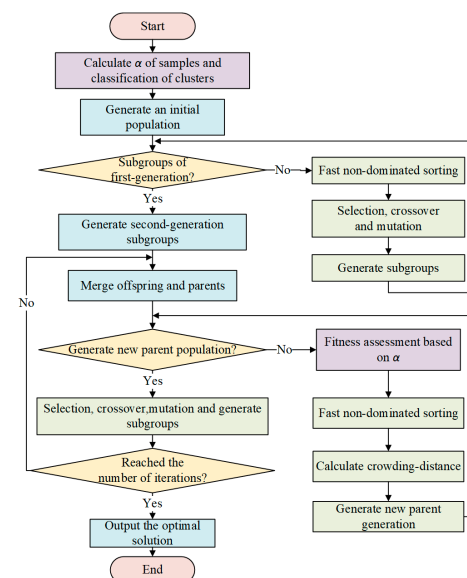


Figure 9. Process of the NSGA-II-based desensitization resource optimization method.

When solving this problem, the population size is set to 20 individuals per generation. This value is determined based on a sensitivity analysis (see Section 6.2), which evaluates population sizes of 20, 30, and 40. The results indicate that a population size of 20 achieves a favorable balance between solution quality and computational efficiency, while maintaining sufficient diversity for effective Pareto front exploration.

Based on the objective functions defined in Section 4.2, the fitness function of each individual is expressed as

$$Fit(i) = (F_1(i), F_2(i), F_3(i)) \quad (14)$$

where F_1 represents the total estimated desensitization execution time, F_2 represents the total computational resource consumption, and F_3 represents the task-group matching

cost. In each generation, individuals are evaluated using this fitness function. When the maximum number of iterations is reached, the Pareto front is extracted, and the final decision solution is selected from the Pareto-optimal solution.

5. Experiment Design

5.1. Experimental Environment

The experiments are based on Python 3.12 running on a Windows 64-bit operating system with 32 GB of onboard RAM, Intel(R) Core(TM) Ultra 5 125 H processor up to 1.20 GHz, number of cores 14, and the experiments run without GPU dependency.

5.2. Experimental Datasets

A self-built desensitization system dataset is adopted in this study, which is derived from the operational environment of a real-world business platform, ensuring the authenticity and integrity of the data. The dataset consists of 300 task records, each corresponding to a single file desensitization operation. The original data are stored in CSV (Comma-Separated Values) log files exported from the desensitization system.

The dataset includes multiple task-related attributes, such as file name, total file size (Byte), total number of rows, number of rows per file partition, number of file partitions, total number of threads, and desensitization execution time (ms), among others. These features comprehensively characterize both data scale and computational resource allocation during the desensitization process. The dataset covers 30 consecutive days of system operation, ensuring temporal representativeness.

To ensure data quality and suitability for subsequent analysis, several preprocessing steps were performed. First, duplicate records were examined, and no duplicates were found in the dataset. Second, timestamp-related fields were converted into elapsed time in milliseconds to maintain consistency in time-based measurements. Third, logical consistency checks were conducted to validate key relationships among features (e.g., the product of the number of slices and the number of rows per slice is approximately equal to the total number of rows, within a tolerance of 5%).

No missing values were present in the original dataset. For outlier detection, the interquartile range (IQR) method was employed, where samples falling beyond $1.5 \times \text{IQR}$ from the first (Q1) or third quartile (Q3) were identified as outliers. A total of four outlier samples (approximately 1.3% of the dataset) were detected, primarily corresponding to tasks with unusually long execution times caused by system interruptions. Instead of removing these samples, they were retained to preserve the realism of the dataset, and their potential impact on the results is discussed in the experimental analysis.

Finally, min-max normalization was applied to all continuous features, including file size, row counts, and execution time, scaling their values to the range [0, 1]. This normalization ensures balanced feature contributions in subsequent distance-based computations and improves the stability of the optimization process.

5.3. Evaluation Metrics for Multi-Objective Optimization Comparison

To comprehensively evaluate the optimization effectiveness and computational efficiency of different multi-objective optimization algorithms, four quantitative performance metrics together with a composite ranking criterion are adopted.

- (1) Runtime (s): The wall-clock execution time required by the optimization algorithm, reflecting its computational overhead.
- (2) Best Total Time: The cumulative desensitization execution time of all tasks under the best resource allocation strategy obtained by each algorithm. This is the primary optimization objective of the proposed framework.

- (3) **Best Total Slices:** The total number of file partitions allocated under the optimal strategy. This metric reflects the amount of partitioning resources required to achieve the corresponding optimization performance.
- (4) **Best Total Threads:** The total number of processing threads allocated under the optimal strategy, indicating the level of computational resources consumed by the optimized solution.
- (5) **Composite Rank Score:** Each algorithm is independently ranked according to the four evaluation metrics (Best Total Time, Best Total Slices, Best Total Threads, and Runtime). The individual ranking positions are then summed with equal weights to obtain the composite rank score. A lower composite rank score indicates better overall optimization performance by jointly considering optimization quality and computational efficiency.

Collectively, these evaluation metrics assess both the optimization effectiveness and the computational cost of different algorithms for the desensitization resource allocation problem.

6. Experimental Results and Analysis

6.1. Hyperparameter Sensitivity Analysis

To assess the robustness of the proposed method and to provide guidance for parameter selection in practical deployment, a systematic sensitivity analysis was conducted on four key hyper-parameters: crossover probability (p_c), mutation probability (p_m), number of generations (G), and population size (N). A one-factor-at-a-time (OFAT) experimental design was adopted, wherein each parameter was varied across three levels while holding the remaining parameters fixed at their baseline values ($p_c = 0.7$, $p_m = 0.2$, $G = 8$, $N = 20$). Each configuration was executed on the full desensitization dataset, and the following metrics were recorded: mean runtime per run (in seconds), mean best total time, mean best total slices, mean best total threads, and a composite rank score that aggregates solution quality across Pareto fronts.

6.1.1. Experimental Design

Table 2 summarizes the results of the sensitivity analysis.

Table 2. Sensitivity analysis of the proposed method hyperparameters.

Group	Value	Mean Runtime Seconds	Mean Best Total Time	Mean Best Total Slices	Mean Best Total Threads	Mean Best Rank Score
crossover_prob	0.6	2.5938	7,356,128.3030	149,025	222,358	4,487,953.5818
crossover_prob	0.7	2.5932	7,353,587.3296	149,022.5	222,362.5	4,486,429.3978
crossover_prob	0.8	2.6345	7,354,741.1617	149,100	222,358	4,487,136.297
generations	8	2.5922	7,353,587.3296	149,022.5	222,362.5	4,486,429.3978
generations	12	3.7186	7,349,604.9675	148,947.5	222,372.5	4,484,026.9805
generations	16	4.8522	7,348,022.9073	148,944.5	222,376	4,483,077.8444
mutation_prob	0.1	2.0162	7,364,112.1218	149,025	222,358	4,492,743.8731
mutation_prob	0.2	2.8606	7,353,587.3296	149,022.5	222,362.5	4,486,429.3978
mutation_prob	0.3	3.1522	7,350,580.8208	148,945.5	222,362	4,484,609.9925
population_size	20	2.1406	7,351,221.4785	148,947.5	222,359	4,484,994.1871
population_size	30	3.2323	7,352,362.9368	148,951.5	222,360.5	4,485,680.1621
population_size	40	4.3135	7,352,491.5322	148,946.5	222,369	4,485,758.0193

The tested parameter ranges were selected to span the practically relevant operating region of genetic algorithms in engineering optimization contexts. Specifically:

- (1) **Crossover probability (p_c):** Tested at 0.6, 0.7, and 0.8. Values below 0.6 were excluded as they would excessively slow convergence, while values above 0.8 risk premature convergence.

- (2) Mutation probability (p_m): Tested at 0.1, 0.2, and 0.3. The lower bound ensures sufficient exploration of the search space, while the upper bound prevents the algorithm from degenerating into random search.
- (3) Number of generations (G): Tested at 8, 12, and 16. These correspond to short, moderate, and extended evolutionary horizons in the context of the dataset scale.
- (4) Population size (N): Tested at 20, 30, and 40 individuals. The range balances solution diversity against computational cost.

6.1.2. Results and Discussion

Crossover Probability. As shown in Table 2, the crossover probability exhibits a relatively mild influence on all metrics. The rank score varies by less than 0.06% across the three tested values, with $p_c = 0.7$ achieving the lowest rank score (4, 486, 429.40) and $p_c = 0.6$ producing marginally worse results (4, 487, 953.58). The mean runtime remains nearly constant (2.59–2.63 s) regardless of p_c , indicating that crossover operations constitute a negligible fraction of the total computational cost. These results suggest that the proposed method is robust to crossover probability variations within the 0.6–0.8 range, with $p_c = 0.7$ recommended as the default setting for the best balance between exploration and exploitation.

Mutation Probability. The mutation probability has a more pronounced effect on both optimization quality and runtime. Increasing p_m from 0.1 to 0.3 reduces the best total time from 7.36×10^6 to 7.35×10^6 , and the rank score by 0.18% (from 4, 492, 743.87 to 4, 484, 609.99). However, mean runtime also increases from 2.02 s to 3.15 s—a 56% increase—due to the higher computational overhead of more frequent mutation operations. The intermediate value $p_m = 0.2$ achieves a favorable balance, offering 99.8% of the rank score improvement at 71% of the runtime increase. Setting $p_m = 0.1$ (lowest runtime but poorest optimization) is only advisable in severely time-constrained scenarios.

Number of Generations. The number of generations shows a clear trade-off between solution quality and computational cost. Extending G from 8 to 16 reduces the best total time from 7.35×10^6 to 7.35×10^6 and the rank score by 0.075% (from 4, 486, 429.40 to 4, 483, 077.84). However, mean runtime scales approximately linearly with G : from 2.59 s at $G = 8$ to 3.72 s at $G = 12$ (+43.5%) and 4.85 s at $G = 16$ (+87.2%). The diminishing returns in solution quality beyond $G = 8$ —combined with the near-linear growth in runtime—indicate that $G = 8$ is the most cost-effective configuration. This finding is also consistent with the convergence trend observed in Figure 10, where the objective values become stable during the later evolutionary stage.

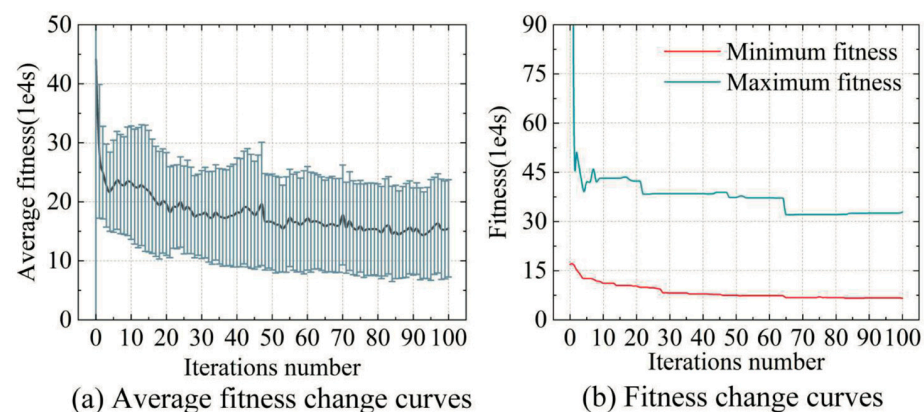


Figure 10. Individual Adaptation Change Curve.

Population Size. Increasing the population size from 20 to 40 results in a slight degradation of solution quality: the rank score increases from 4, 484, 994.19 (at $N = 20$)

to 4, 485, 758.02 (at $N = 40$), and the Best Total Time rises from 7.35×106 to 7.35×106 . Simultaneously, mean runtime doubles from 2.14 s to 4.31 s. This counter-intuitive result—where a larger population does not improve optimization outcomes—can be explained by the nature of the problem: with a limited number of decision variables (slices, threads, and algorithm assignment) and the strong guidance provided by the sensitivity coefficient α and task group matching cost θ , a modest population of 20 individuals is sufficient to explore the Pareto front effectively. Larger populations introduce additional search diversity. Under a fixed evolutionary budget ($G = 8$), this increased diversity reduces the number of effective evolutionary updates available for convergence, resulting in only marginal improvements in solution quality while increasing computational cost.

Recommended Configuration. Based on the sensitivity analysis, the recommended default hyperparameter configuration for the proposed method is $N = 20$, $G = 8$, $p_c = 0.7$, and $p_m = 0.2$. This configuration achieves the best balance between optimization quality, runtime efficiency, and result stability across all four evaluated dimensions. It is worth noting that the rank score is remarkably stable across all 12 tested configurations (coefficient of variation $< 0.1\%$), demonstrating the inherent robustness of the proposed method to hyperparameter perturbations—a desirable property for real-world deployment where fine-tuning may be impractical.

6.2. Comparative Evaluation with Mainstream Multi-Objective Optimization Algorithms

To further validate the effectiveness of the proposed method, a comprehensive comparative experiment was conducted against five representative multi-objective optimization algorithms widely adopted in the literature: NSGA-II [23], NSGA-III [24], MOEA/D [25], SPEA2 [26], and MOPSO [27]. All algorithms were evaluated on the same self-built desensitization system dataset comprising 30 days of operational data from a real-world business platform.

6.2.1. Experimental Setup

To ensure a fair comparison, all algorithms were configured with consistent parameter settings. The population size was set to 20 for all methods. For the proposed method, the crossover probability was set to 0.7, the mutation probability to 0.2, and the number of generations to 8 (the optimal configuration identified through sensitivity analysis, see Section 6.2). For MOEA/D, the neighborhood size was set to 10 and the number of weight vectors to 20. For NSGA-II and NSGA-III, crossover and mutation probabilities were set to 0.7 and 0.2, respectively, with 8 generations. SPEA2 employed an archive size of 20 with identical genetic operator settings. MOPSO used 20 particles with inertia weight $w = 0.7298$ and cognitive/social coefficients $c_1 = c_2 = 1.4962$. Each algorithm was executed for 10 independent runs to obtain statistically meaningful results, and the mean and standard deviation of each metric were recorded.

6.2.2. Results and Analysis

Table 3 presents the comparative results.

Table 3. Repeated-run comparison (mean $\pm$ std) of methods on the file optimization benchmark.

Mthod	Runtime(s)	Best Total Time	Best Total Slices	Best Total Threads
The proposed method	2.4897 $\pm$ 0.0106	7,354,943.3373 $\pm$ 5547.8483	148,996 $\pm$ 90.27	222,362 $\pm$ 6.93
MOEA/D	0.3999 $\pm$ 0.01	87,037,427.1041 $\pm$ 8,333,447.0918	76,407.33 $\pm$ 3018.65	112,687.33 $\pm$ 746.80
NSGA-II	0.1654 $\pm$ 0.0009	95,963,817.3878 $\pm$ 2,135,430.7373	76,680.33 $\pm$ 145.73	114,065.67 $\pm$ 1349.80
NSGA-III	0.1685 $\pm$ 0.0023	97,808,667.9557 $\pm$ 8,098,980.9481	72,915 $\pm$ 327.41	110,932 $\pm$ 2089.86
SPEA2	0.19 $\pm$ 0.0063	98,650,017.7807 $\pm$ 14,396,004.7547	74,783 $\pm$ 1733.95	113,297 $\pm$ 4286.39
MOPSO	1.1949 $\pm$ 0.255	136,679,455.2208 $\pm$ 717,129.3464	75,862 $\pm$ 995.51	114,697.67 $\pm$ 1309.14

The proposed method achieves a best total time of 7.35×10^6 , which is approximately $11.8\times$ lower than MOEA/D (87.04×10^6) and up to $18.6\times$ lower than MOPSO (136.68×10^6). The standard deviation of the proposed method ($\pm 5.55 \times 10^3$) is also substantially smaller than those of the competing methods, indicating superior stability. The superior optimization performance can be attributed to the incorporation of the personalized sensitivity coefficient and task-group modeling, which enable the optimization process to exploit problem-specific characteristics during resource allocation. It should be noted that the proposed method incorporates domain-specific information through the personalized sensitivity coefficient and task-group modeling, whereas the baseline algorithms are applied as general-purpose optimization methods without problem-specific adaptations. Therefore, the comparison aims to demonstrate the effectiveness of integrating domain knowledge into the optimization framework rather than to compare optimization algorithms in isolation.

Resource Allocation Strategy. The best total slices and best total threads metrics reveal distinct resource allocation behaviors. The proposed method allocates substantially more slices (1.49×10^5) and threads (2.22×10^5) compared to competing methods (slices: $0.73\text{--}0.77 \times 10^5$; threads: $1.08\text{--}1.15 \times 10^5$). This indicates that the proposed method adopts an aggressive parallelization strategy—fine-grained file partitioning combined with high thread concurrency—which, while increasing resource consumption, dramatically reduces the overall desensitization execution time. In contrast, the competing algorithms converge toward conservative resource configurations that fail to fully exploit the parallelism potential of the system. Such a strategy is particularly suitable for offline desensitization scenarios where execution time is prioritized over temporary resource occupancy.

Runtime Overhead. The runtime of the proposed method (2.49 s) is higher than those of the baseline algorithms because additional computations, including personalized sensitivity coefficient calculation and task-group matching evaluation, are performed during each evolutionary generation. Nevertheless, the additional optimization overhead is only a few seconds and is negligible compared with the substantial reduction in desensitization execution time achieved by the optimized resource allocation strategy. Since the optimization process is performed offline before task execution, this additional computational cost is acceptable for practical deployment.

Comparison among baseline algorithms. Among the baseline algorithms, MOEA/D achieves the best optimization performance, followed by NSGA-II, NSGA-III, SPEA2, and MOPSO. This observation indicates that decomposition-based evolutionary optimization is more suitable for the present resource allocation problem than particle swarm optimization. Nevertheless, all baseline algorithms are consistently outperformed by the proposed method owing to the incorporation of task-specific knowledge into the optimization framework.

In summary, the comparative experiment demonstrates that the proposed method, by integrating domain-specific knowledge (sensitivity coefficients and algorithm-task clustering) into the NSGA-II framework, significantly outperforms generic multi-objective optimization algorithms in terms of solution quality, while incurring only a modest and acceptable computational overhead. The low standard deviations across all metrics further confirm the robustness and reliability of the proposed method.

6.3. Visualization and Analysis of Optimization Results

6.3.1. Convergence Analysis

The optimal solution is obtained by solving the multi-objective optimization problem using the NSGA-II-based optimization method, and the optimization results are further analyzed.

For visualization purposes, the convergence behavior of the proposed algorithm is illustrated using an extended optimization run of 100 generations, whereas the hyperparameter sensitivity analysis in Section 6.2 identifies $G = 8$ as the recommended default configuration considering the trade-off between optimization quality and computational efficiency. Figure 10 demonstrates the convergence behavior of the proposed optimization framework. During the early evolutionary stage, the average fitness fluctuates considerably because the initial population is randomly generated and contains many suboptimal resource allocation strategies. As the evolutionary process proceeds, non-dominated sorting and elitist selection gradually eliminate inferior individuals, resulting in a steady decrease in the average execution time and convergence of the population. The convergence observed after approximately the 70th iteration indicates that the proposed optimization framework can rapidly identify stable Pareto-optimal solutions without excessive evolutionary iterations.

6.3.2. Pareto Solution and Task-Group Analysis

Figures 11 and 12 illustrate the optimization process from two complementary perspectives. Figure 11 presents the Pareto-optimal solution set generated by NSGA-II, while Figure 12 shows the corresponding redistribution of tasks among different task groups after optimization. Compared with the initial clustering results in Figure 7b, the optimized task groups exhibit a more balanced distribution, indicating that the proposed optimization framework effectively reallocates computational resources by matching tasks with more suitable task groups. This balanced distribution contributes directly to the reduction in execution time observed in the subsequent experiments.

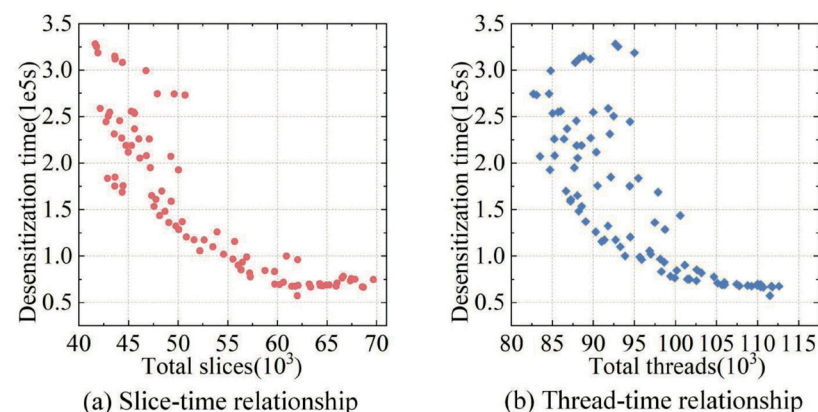


Figure 11. Pareto Frontier Solution Set (100 rounds of iterations).

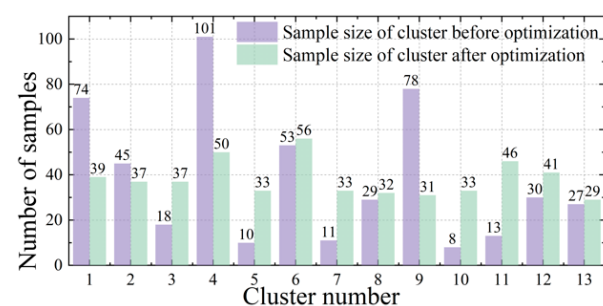


Figure 12. Comparison of clustering distribution after optimization.

6.3.3. Resource Allocation Analysis

Figures 13–15 illustrate the changes in execution time, slice allocation, and thread allocation before and after optimization. As shown in Figure 13, the proposed method successfully reduces the execution time for 92.76% of the desensitization tasks, demonstrating

the effectiveness of the optimized resource allocation strategy. Figures 14 and 15 further reveal that the reduction in execution time is achieved through coordinated adjustment of both slice allocation and thread allocation, rather than by modifying a single resource variable. Although the average numbers of slices and threads increase after optimization, the additional resource consumption is compensated by the substantial reduction in execution time. Moreover, 31.59% of the tasks achieve shorter execution time while requiring fewer thread resources than the original configuration, indicating that the proposed optimization framework can simultaneously improve execution efficiency and resource utilization for a considerable proportion of tasks.

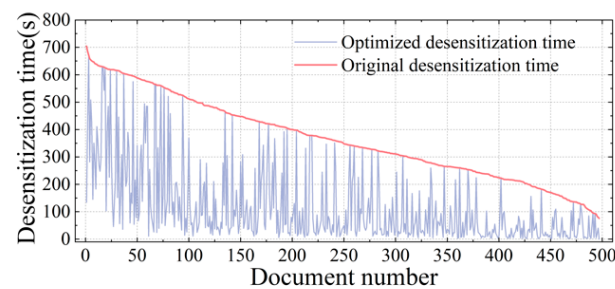


Figure 13. Comparison of Desensitization Time Consumption.

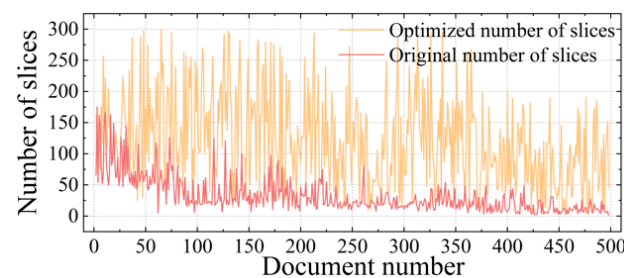


Figure 14. Comparison of the number of slices.

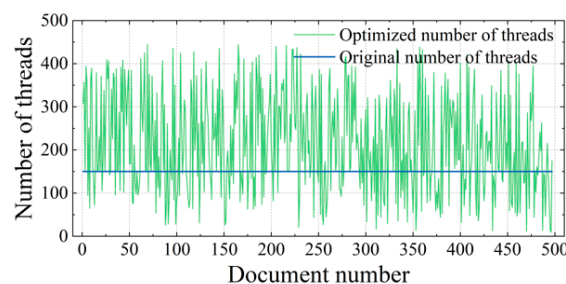


Figure 15. Comparison of thread counts.

6.3.4. Resource–Performance Relationship

Figure 16 further illustrates the relationship between resource allocation and execution time after optimization. Compared with the original distributions shown in Figures 2 and 5, the optimized samples become more concentrated in the low-execution-time region. Although a small proportion of tasks still require additional thread resources, these tasks generally correspond to larger files or higher-complexity desensitization tasks, whose execution time can be significantly reduced through increased parallelism. This observation confirms that the proposed optimization framework dynamically balances resource consumption and execution efficiency according to task characteristics.

Overall, the visualization results consistently demonstrate that the proposed optimization framework does not rely on a single optimization variable. Instead, execution time reduction is achieved through the coordinated optimization of slice allocation, thread

allocation, and task-group assignment. The convergence analysis, Pareto-front distribution, and resource allocation results collectively verify that the proposed method effectively balances execution efficiency and computational resource consumption under heterogeneous desensitization requirements.

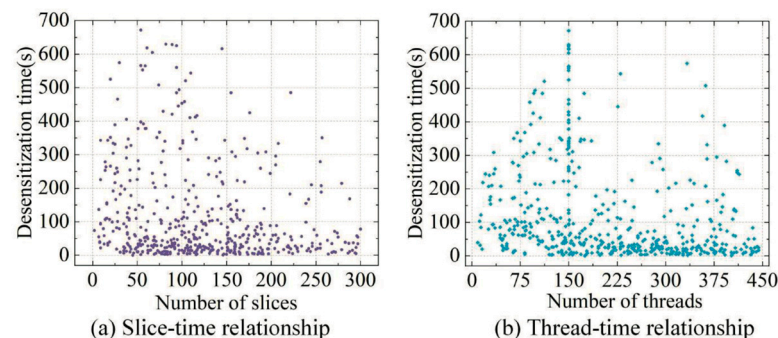


Figure 16. Optimized sample distribution.

7. Conclusions and Future Work

This paper proposes a multi-objective optimization method for desensitization resource allocation of civil aviation passenger sensitive data under resource-constrained environments. To address the limited information available in the original desensitization system logs, particularly the fixed thread configuration and the absence of explicit algorithm labels, a mathematical model for estimating desensitization execution time is developed. The proposed model integrates a personalized sensitivity coefficient with K-means-based task grouping to characterize heterogeneous computational characteristics of desensitization tasks. Based on this model, an NSGA-II-based multi-objective optimization framework is constructed to jointly optimize desensitization execution time, computational resource consumption, and task-group matching cost.

Experimental results demonstrate that the proposed method effectively improves resource allocation efficiency. Specifically, 92.76% of the desensitization tasks are successfully optimized, while 31.59% of the tasks achieve shorter execution time with fewer thread resources than the original configuration. Furthermore, hyperparameter sensitivity analysis confirms the robustness of the proposed framework, and comparative experiments with representative multi-objective optimization algorithms demonstrate its superior optimization performance.

The experimental results indicate that integrating task-group modeling with personalized sensitivity analysis enables more effective resource allocation than conventional optimization methods that rely solely on generic evolutionary search. Nevertheless, several limitations remain. First, the proposed task groups are generated through unsupervised K-means clustering and therefore represent computational task categories rather than actual desensitization algorithm labels. Second, the proposed optimization framework is validated using operational data collected from a single desensitization platform, and its generalizability to other application scenarios requires further investigation. Finally, the current model focuses primarily on resource allocation efficiency and does not explicitly evaluate the effectiveness of different desensitization rules in terms of privacy protection.

Future work will focus on incorporating additional operational features and real-time system information into the optimization framework, investigating more advanced task-group construction methods, and extending the proposed approach to adaptive online resource allocation in dynamic computing environments.

Author Contributions: Conceptualization, methodology, experiment, S.W. and H.Z.; validation, J.Z.; experimental analysis, S.W.; data analysis, S.W. and Z.L.; writing—original draft preparation, S.W.;

writing—review and editing, L.D.; conceptualization, project administration, funding acquisition, Z.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the fundamental research funds for the central universities under grant no. 3122025054.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: This study did not involve human subjects, animal experiments, or any direct collection of personal data from individuals. The experimental dataset consists solely of machine-generated system operation logs recording computational resource metrics (file sizes, processing times, thread counts) from an automated desensitization system. No individual-level personal information, biometric data, or any data that could identify specific passengers is present in the dataset. The anonymous system logs were collected as part of routine IT infrastructure monitoring and performance optimization activities. Therefore, in accordance with institutional policies and national regulations (including China's Personal Information Protection Law), this study does not meet the criteria requiring ethics committee review or informed consent.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: During the preparation of this manuscript, the author used ChatGPT-3.5 to improve the language and readability of the manuscript. The author reviewed and edited the generated content and takes full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Shah, I.A.; Jhanjhi, N.Z.; Brohi, S. Cybersecurity issues and challenges in civil aviation security. In *Cybersecurity in the Transportation Industry*; Wiley Online Library: Hoboken, NJ, USA, 2024; pp. 1–23.
2. Zhang, L.L.; Cao, W.D.; Wang, H.C. A multi-user dynamic symmetric searchable encryption scheme supporting conjunctive search. *J. Comput. Res. Dev.* **2022**, *59*, 2309–2322.
3. Li, R. Research and Application of Data Dynamic Masking Technology. Master's Thesis, Department of Electronics, Beijing Jiaotong University, Beijing, China, 2023.
4. Zhang, Y.L. Research and Implementation of Desensitization Publishing Technology for Multimodal Data. Master's Thesis, Department of Engineering, Xi Dian University, Xi'an, China, 2022.
5. Lv, X.Y. Research on Desensitization and Optimization of Medical Text Based on SNOMED-CT. Master's Thesis, Department of Engineering, Tianjin University of Technology, Tianjin, China, 2022.
6. Liu, Z.P.; Sun, J.W.; Wang, S.; Wang, W.S.; Yin, W.Z.; Zhang, B. PDMP: ϵ k personalized data masking protection. *Appl. Res. Comput.* **2020**, *37*, 3068–3082.
7. Yang, R.; Gao, H.; Si, F.; Wang, J. Intelligent Scene-Adaptive Desensitization: A Machine Learning Approach for Dynamic Data Privacy in Virtual Power Plants. *Electronics* **2024**, *13*, 1051. [[CrossRef](#)]
8. Jiang, L.; Zhang, L.; Tan, S.; Li, D.; Hwang, C.; Fan, J.; Li, E.P. A novel physics-assisted genetic algorithm for decoupling capacitor optimization. *IEEE Trans. Microw. Theory Tech.* **2024**, *72*, 4498–4507. [[CrossRef](#)]
9. Li, Y.; Luo, X. Adaptive synthesis using hybrid genetic algorithm and particle swarm optimization for reflectionless filter with lumped elements. *IEEE Trans. Microw. Theory Tech.* **2023**, *71*, 5317–5334. [[CrossRef](#)]
10. Nguyen, S.; Thiruvady, D.; Zhang, M.; Alahakoon, D. Automated design of multipass heuristics for resource-constrained job scheduling with self-competitive genetic programming. *IEEE Trans. Cybern.* **2021**, *52*, 8603–8616.
11. Li, M.; Ma, H.; Lv, S.; Wang, L.; Deng, S. Enhanced NSGA-II-based feature selection method for high-dimensional classification. *Inf. Sci.* **2024**, *663*, 120269. [[CrossRef](#)]
12. Jiwapatria, S.; Setio, H.D.; Sidi, I.D.; Kusumaningrum, P. Multi-objective optimization of active control system using population guidance and modified reference-point-based NSGA-II. *Results Control Optim.* **2024**, *16*, 100453. [[CrossRef](#)]
13. Akopov, A.S.; Beklaryan, L.A.; Thakur, M. Improvement of maneuverability within a multiagent fuzzy transportation system with the use of parallel biobjective real-coded genetic algorithm. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 12648–12664. [[CrossRef](#)]
14. Maroof, A.; Ayvaz, B.; Naeem, K. Logistics optimization using hybrid genetic algorithm (hga): A solution to the vehicle routing problem with time windows (vrptw). *IEEE Access* **2024**, *12*, 36974–36989. [[CrossRef](#)]

15. Zhang, H.; Miao, C.; Zhu, W.; Hu, Y.; He, J. Optimization of high-speed channel for signal integrity with deep genetic algorithm. *IEEE Trans. Compon. Packag. Manuf. Technol.* **2022**, *12*, 1161–1171.
16. Vijai, P. A hybrid multi-objective optimization approach with NSGA-II for feature selection. *Decis. Anal. J.* **2025**, *14*, 100550. [[CrossRef](#)]
17. Jiang, Q.; Wang, P. NSGA-II algorithm based control parameters optimization strategy for megawatt novel nuclear power systems. *Energy* **2025**, *316*, 134444. [[CrossRef](#)]
18. Gu, Z.R.; Ding, J.Y.; Chai, Z. TC-NSGA-II: Multi-Objective Optimization Algorithm for TSN-CAN Gateway Congestion Scheduling. *J. Beijing Univ. Aeronaut. Astronaut.* **2025**, *69*, 1–17. [[CrossRef](#)]
19. MacQueen, J. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*; University of California Press: Oakland, CA, USA, 1967; Volume 1, pp. 281–297.
20. Lloyd, S.P. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **1982**, *28*, 129–137. [[CrossRef](#)]
21. Doerr, B.; Ivan, T.; Krejca, M.S. Speeding up the NSGA-II with a simple tie-breaking rule. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Philadelphia, PA, USA, 25 February–4 March 2025; Volume 39, pp. 26964–26972.
22. Hu, J.; Luo, H.; Chen, Y.; Wang, G.; Xiao, X.; Cai, Z. Adaptive NSGA-II algorithm for integrated production-distribution scheduling in multi-plant regional collaborative manufacturing systems. *Comput. Integr. Manuf. Syst.* **2026**, *32*, 1486–1500.
23. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [[CrossRef](#)]
24. Deb, K.; Jain, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. *IEEE Trans. Evol. Comput.* **2014**, *18*, 577–601. [[CrossRef](#)]
25. Zhang, Q.; Li, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731. [[CrossRef](#)]
26. Zitzler, E.; Laumanns, M.; Thiele, L. SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK-Rep.* **2001**, *103*, 1–21.
27. Coello Coello, C.A.; Pulido, G.T.; Lechuga, M.S. Handling multiple objectives with particle swarm optimization. *IEEE Trans. Evol. Comput.* **2004**, *8*, 256–279. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.