*Journal of Network Intelligence*
**Taiwan Ubiquitous Information**
 ISSN 2414-8105
Volume **10**, Number **4**, November **2025**

# Hadoop Enterprise Production Task Scheduling Based on QoS Constraints and Frog Hopping Algorithm

Jin-Xiang Peng

School of Information Engineering
Hunan Applied Technology University
Changde 415000, P. R. China
821296101@qq.com

Li Zhang*

School of Information Engineering
Hunan Applied Technology University
Changde 415000, P. R. China
67260282@qq.com

Shao-Qiang Ye

Faculty of Computing
Universiti Teknologi Malaysia
Skudai 81310, Malaysia
shaoq_ye@163.com

*Corresponding author: Li Zhang
Received July 11, 2024, revised December 3, 2024, accepted August 14, 2025.

Abstract. *Traditional production task scheduling approaches struggle to effectively cope with resource dynamic changes and quality assurance service (QoS) constraints when facing large-scale distributed environments, especially in enterprise applications on the Hadoop platform. To address this challenge, this study proposes an innovative production task scheduling framework that combines QoS constraints and an improved hybrid Frog Leaping Algorithm (SFLA) to optimise task allocation in Hadoop environments. First, by integrating QoS metrics, it ensures that task scheduling not only considers execution efficiency, but also fully satisfies quality of service requirements such as latency and throughput. Second, SFLA is optimised for the characteristics of Hadoop clusters by introducing adaptive hop-step length adjustment and elite strategy, which enhances the exploration and exploitation capabilities of the algorithm, thus improving the flexibility and adaptability of the scheduling strategy. Experimental results show that compared with other heuristic algorithms such as Genetic Algorithm (GA) and Particle Swarm Optimisation (PSO), the improved SFLA algorithm reduces the solution time while ensuring that a high-quality QoS scheduling solution is found, which is particularly important for cloud manufacturing environments where efficiency and accuracy are pursued together.*
**Keywords:** Hadoop platform; production task scheduling; QoS constraints; SFLA

1. **Introduction.** Production task scheduling, as a core aspect of modern enterprise operation [1, 2], is directly related to the optimisation of production efficiency and resource utilisation, and its importance is self-evident. In the complex and changing production environment, effective scheduling can ensure that the production tasks are completed on time, and at the same time balance the allocation of resources to avoid bottlenecks, reduce

production costs, and enhance overall competitiveness. In the face of increasing market demand and production flexibility requirements, how to achieve efficient scheduling of production tasks under limited resources and time constraints has become a key issue to be solved [3]. Especially in the context of the booming development of cloud computing and big data technology, the research on production task scheduling has risen to a new height, aiming to achieve the fine management of the production process by means of intelligence and automation to meet the ability to respond quickly to the market changes and customer demand [4]. Therefore, exploring advanced production task scheduling strategies and technical means has important practical value and theoretical significance for improving production efficiency, reducing costs, and enhancing the market adaptability of enterprises.

The rise of cloud platforms brings unprecedented application potential for production task scheduling. As a typical representative of resource pooling and on-demand services, cloud platforms are able to integrate distributed computing resources, providing powerful support for dynamic allocation and optimisation of production tasks. Through centralised management and intelligent scheduling in the cloud, enterprises can flexibly respond to fluctuations in production tasks and achieve efficient reuse and dynamic adjustment of resources [5]. The elastic scalability of cloud platforms ensures that resources can be quickly expanded to handle unexpected tasks during peak hours, while resources can be released to reduce idleness during trough periods, thus maximising resource utilisation. In addition, the widely interconnected nature of cloud platforms facilitates inter-enterprise collaborative operations, making cross-geographical and cross-enterprise scheduling of production tasks possible, which further enhances the responsiveness and collaborative efficiency of the supply chain [6]. In summary, the application of cloud platform in production task scheduling can not only achieve the intelligence and automation of the production process, but also effectively improve the flexibility and economy of the production system under the premise of guaranteeing the QoS, showing great application potential and optimisation space.

1.1. **Related work.** Production task scheduling, as a key link to optimise the utilisation of manufacturing resources and improve production efficiency, has been a hot research topic in the field of industrial engineering and operations management. Early scheduling methods mainly relied on traditional operations research models and algorithms, such as the obsolete First-In-First-Out (FIFO) [7], Min-Min [8], Max-Min [9] and Round Robin [10] scheduling rules. For example, Murad et al. [11] explored the application of Min-Min based algorithm in pipelined job scheduling, although this algorithm is easy to implement, it does not take into account the inter-task dependencies enough and is inefficient in handling large-scale tasks.

As the complexity of the problem increased, scholars began to explore more advanced optimisation techniques. Meta-heuristic algorithms such as Genetic Algorithm (GA), Particle Swarm Optimisation (PSO), and Simulated Annealing (SA) are widely used in production task scheduling. For example, Ishikawa et al. [12] proposed an improved genetic algorithm to solve the Flexible Job Shop Scheduling Problem (FJSP), which significantly improves the algorithm's convergence speed and the quality of the solution through an innovative population initialisation strategy and adaptive cross-variation probabilities. Although these algorithms improve the scheduling results to some extent, they are usually difficult to deal with highly dynamic and uncertain production environments and are prone to fall into local optima.

In recent years, with the rapid development of cloud computing and big data technologies [13, 14], cloud manufacturing platforms have become a new focus of production task

scheduling research. Under the cloud manufacturing environment, dynamic sharing of resources, remote execution of tasks, and massive processing of data bring new challenges and opportunities for production task scheduling. Currently, the research on production task scheduling under cloud manufacturing platform mainly focuses on the following directions:

1. **Hybrid task scheduling model:** considering the diversity of tasks in cloud environments, scholars have worked on developing hybrid scheduling models applicable to both independent tasks and workflow tasks. For example, Zhou et al. [15] proposed an improved genetic algorithm for the problem of low resource utilisation, which incorporates Min-Min and Max-Min strategies to improve the global efficiency of task scheduling. Although these studies optimise resource allocation to a certain extent, they are still inadequate in dealing with QoS constraints and dynamic changes in task execution.
2. **QoS-aware scheduling algorithms:** with the improvement of users' requirements on quality of service (QoS), how to comprehensively consider multidimensional QoS constraints, such as time, cost, and reliability, in scheduling process has become a research focus. For example, Li et al. [16] proposed a task scheduling algorithm with multiple QoS and trust by fuzzy clustering method, which improved the trust and reliability of task scheduling. However, the real-time and flexibility of these algorithms in handling large-scale and highly dynamic tasks still need to be improved.
3. **Introduction of Intelligent Optimisation Algorithms:** in view of the limitations of traditional algorithms in dealing with large-scale and complex problems, more and more researches have begun to explore the innovative applications of intelligent algorithms. For example, Meyyappan [17] proposed a wavelet neural network model based on SFLA (Shuffled Frog Leaping Algorithm), which aims to improve the accuracy of power load forecasting, and although this research is not in the direct scope of cloud manufacturing, it demonstrates the potential of intelligent algorithms in dealing with complex optimisation problems.

1.2. **Motivation and contribution.** Existing production task scheduling strategies in cloud computing environments, especially for the Hadoop platform, tend to focus on the efficiency of resource allocation and the optimisation of task execution, while ignoring the dynamics and diversity of user-specific quality of service (QoS) requirements in task scheduling. Task scheduling on the Hadoop platform faces a number of challenges, including, but not limited to, the rapidly changing nature of resources, the complexity of inter-task dependencies, and the stringent user requirements for task execution time, cost, and success rate. Traditional scheduling algorithms, such as First-Come-First-Served (FCFS) [18] and Priority Scheduling [19], although simple and easy to implement, are difficult to accurately satisfy the QoS constraints imposed by users in the face of large-scale, high-concurrency, and heterogeneous task sets, which often leads to a large deviation of the task scheduling results from the users' expectations.

To address the above problems, this research proposes a Hadoop enterprise production task scheduling framework based on QoS constraints and improved SFLA. The core innovation and contribution of this work is:

1. Aiming at the characteristics of the Hadoop platform, this study incorporates the QoS constraints proposed by users (e.g., task completion time, cost budget and resource utilisation) into the task scheduling model. By introducing a QoS-sensitive scheduling objective function, the model is able to adapt more flexibly to the specific needs of different users, ensuring that the task scheduling results can satisfy the time

constraints as well as control the cost under limited resources, thus improving user satisfaction.

2. Aiming at the complexity of task scheduling in Hadoop platform, this study proposes a scheduling algorithm based on improved SFLA, which integrates local search and global hopping strategies, and effectively avoids the problems of traditional frog-hopping algorithms that are prone to fall into local optimums and inefficient searches by adaptively adjusting the search step lengths and using the historical optimal solutions to guide the search process. The improved SFLA demonstrates excellent global search capability and faster convergence speed when dealing with large-scale task sets, thus significantly improving the efficiency and accuracy of task scheduling while satisfying the QoS constraints.

## 2. Related technical studies.

### 2.1. Hadoop cloud computing manufacturing platform.

Hadoop is an open-source distributed computing framework [20, 21], which has become the cornerstone of the big data processing and cloud computing domains with its highly scalable, fault-tolerant, and cost-effective resource management capabilities, as shown in Figure 1. The Hadoop ecosystem covers data storage (e.g., HDFS), resource management (e.g., YARN), data processing (e.g., MapReduce) and many other key components, providing an all-encompassing solution for massive data processing. In the manufacturing field, the Hadoop platform provides rich data support and powerful computing power for production task scheduling by integrating data from different production lines, sensors, and management systems.

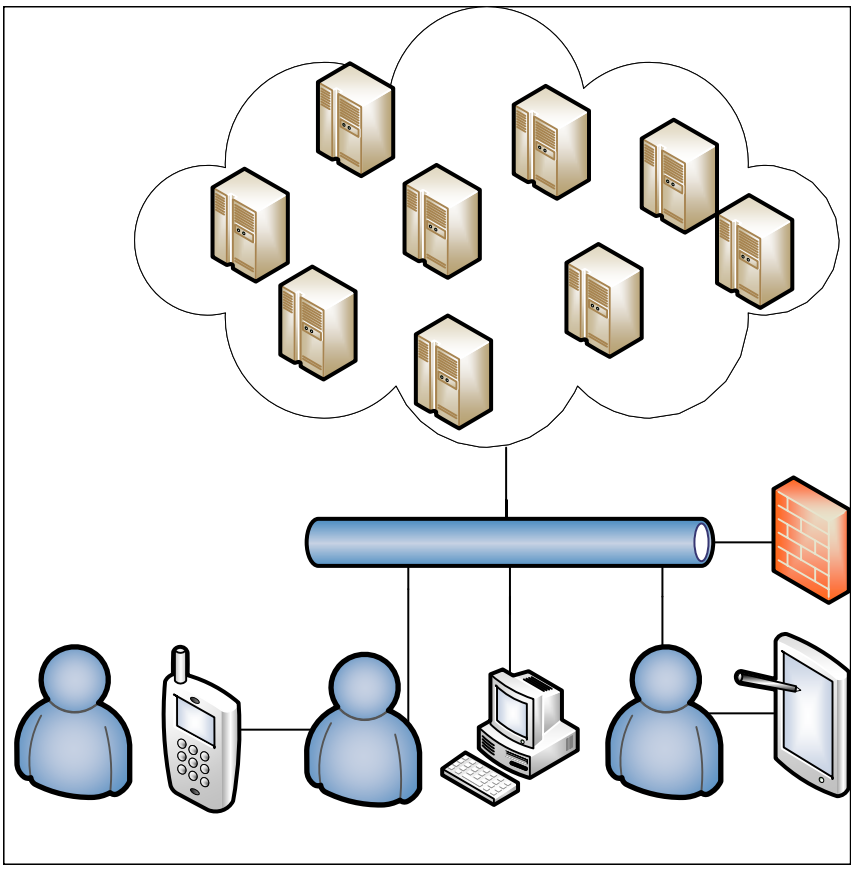

Figure 1. Hadoop Cloud Manufacturing Platform.

The core components of Hadoop [22] include Hadoop Distributed File System (HDFS) and Yet Another Resource Negotiator (YARN). HDFS provides a distributed file system that is capable of handling petabytes of data and ensures high availability and fault tolerance. The principle of HDFS is shown in Figure 2. YARN acts as a resource manager that optimises the allocation of cluster resources, enabling multiple computing frameworks (including, but not limited to, MapReduce and Spark) to coexist and run efficiently on the same cluster. This architectural design enables the Hadoop platform to flexibly respond to the diverse task scheduling needs of enterprise production, including but not limited to large-scale data processing, real-time analysis, and batch processing tasks.

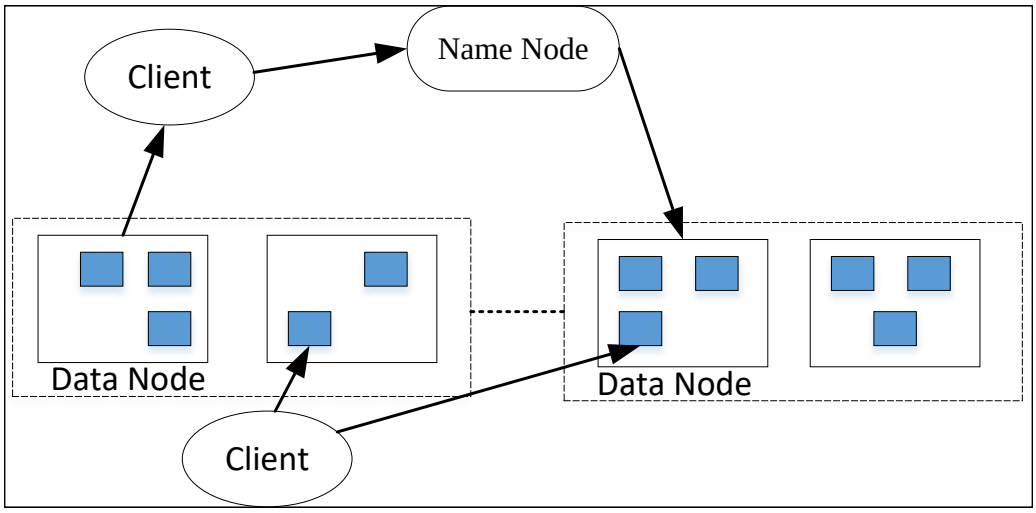


Figure 2. Principles of HDFS.

In a cloud manufacturing environment, the Hadoop platform supports real-time monitoring and dynamic scheduling of production tasks through its powerful data processing capabilities. It is able to process massive data from production tasks, perform rapid analysis, identify production bottlenecks, and optimise resource allocation. In addition, the combination of Hadoop and cloud manufacturing services enables elastic scaling of tasks and dynamic adjustment of computing resources according to the real-time demand of production tasks, thus improving production efficiency and flexibility. The Hadoop cloud computing manufacturing platform provides infrastructure and technical support for achieving efficient and intelligent production task scheduling. Its distributed characteristics, powerful data processing capability and resource management mechanism provide an ideal implementation environment for the study of hybrid task scheduling under satisfying QoS constraints. Next, based on this platform, we will explore how to further optimise the production task scheduling by using the hybrid frog hopping algorithm to ensure that the overall productivity and responsiveness are improved while meeting the QoS requirements.

2.2. **Hybrid Frog Leaping Algorithm Principles.** In an in-depth discussion of the QoS constraint-based task scheduling problem, we introduce SFLA [23, 24], an optimisation method that innovates and improves upon the traditional Frog Leaping Algorithm (FLA), which is particularly suitable for solving optimisation problems in complex production task scheduling. The Hybrid Frog Leaping Algorithm draws on the advantages of other meta-heuristic algorithms, such as PSO and GA, and improves the algorithm's global search capability and local search accuracy by combining the efficient search strategies of these algorithms. In the following, the basic principles of the hybrid frog jump

algorithm will be introduced in detail and its main process will be described through mathematical formulas.

The FLA is inspired by the behaviour of frogs searching for food in ponds in nature, and the frogs in this algorithm represent potential solutions in the solution space, exploring the solution space by "jumping" to find the optimal solution. The traditional FLA mainly consists of two phases, local search and global search, in which the local search aims to find the local optimal solution by exploring the neighbourhood, while the global search allows the frog to cross a large distance to jump out of the local optimal trap and find the global optimal solution.

The hybrid frog jump algorithm adds the following innovations to the original [25]:

**Initialisation with hybrid strategies**: the SFLA algorithm combines random generation, heuristic initialisation based on historical optimal solutions and adaptive initialisation strategies based on task characteristics during initialisation to ensure population diversity.

**Adaptive Jump Step Adjustment:** by introducing an adaptive mechanism to dynamically adjust the frog's jump step $L_i$, the SFLA algorithm can quickly cover the solution space at the beginning of the search and focus on local fine search at the later stage.

$$L_i = L_{\min} + (L_{\max} - L_{\min}) \times e^{-\alpha \cdot t} \tag{1}$$

where $L_{\min}$ and $L_{\max}$ are the minimum and maximum jump steps, respectively; $\alpha$ is the attenuation coefficient; and $t$ denotes the current iteration number.

**Hybrid strategy fusion:** the particle velocity update mechanism in the PSO algorithm and the cross-mutation operation of the GA are combined in the search process as follows:

$$v_{ij}(t+1) = w \cdot v_{ij}(t) + c_1 \cdot r_1 \cdot (pbest_{ij} - x_{ij}(t)) + c_2 \cdot r_2 \cdot (gbest_j - x_{ij}(t)) \tag{2}$$

where $v_{ij}$ represents the velocity of the $i$th frog in dimension $j$; $w$ is the inertia weight; $c_1, c_2$ are the learning factors; $r_1, r_2$ are the random numbers; and $pbest_{ij}$ and $gbest_j$ represent the individual optimal and global optimal solutions, respectively.

The main process of the hybrid frog jump algorithm can be summarised in the following steps:

1. **Initialisation:** generate an initial population represents a possible task scheduling scheme. $X = \{x_1, x_2, ..., x_N\}$, each solution $x_i$.
2. **Evaluation and update:** Calculate the fitness value of each solution and update the individual optimal solution *pbest* and the global optimal solution *gbest* according to the fitness value.
3. **Jumping and updating:** for each frog $x_i$, adjust its position based on

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1),$$

   and jump according to the hybrid strategy.
4. **Local search:** after each frog jump, a local search is performed to optimise the local fitness of the solution.
5. **Iteration and termination:** repeat steps 2 to 4 until a predetermined number of iterations is reached or the convergence criterion is satisfied.

Through the above process, the SFLA algorithm not only inherits the efficient search capability of the traditional FLA algorithm, but also effectively avoids the problems of premature convergence and local optimum by integrating the advantages of other algorithms.

## 3. Modelling the problem.

3.1. **Production task scheduling framework.** In a cloud manufacturing environment, the construction of a production task scheduling framework is a key component to achieve efficient resource utilisation and flexible production. The framework aims to combine the distributed nature of cloud manufacturing with the complexity of production scheduling to ensure smooth execution of production tasks and optimal allocation of resources.

3.1.1. *Overview of the scheduling framework.* The cloud manufacturing production task scheduling framework consists of three core components: task decomposition and matching module, dynamic resource allocation module, and production plan optimisation and execution module. First, the task decomposition and matching module is responsible for converting customer demands into specific production tasks and matching them according to the production capacity of each manufacturing enterprise. Second, the dynamic resource allocation module flexibly allocates manufacturing resources according to the current resource status and task demand to ensure the feasibility of task execution. Finally, the production plan optimisation and execution module generates and executes the most efficient production scheduling plan based on optimisation algorithms, taking into account the priority of tasks, resource availability and QoS requirements.

3.1.2. *Task decomposition and matching.* The module first receives customer orders from the cloud manufacturing platform, and then refines the orders into a series of production tasks based on the product structure and manufacturing process requirements. These tasks need to explicitly specify the processing steps, required resource types and workpiece information. Next, intelligent matching algorithms are used to match the production tasks with idle or soon-to-be-vacant manufacturing resources registered with the cloud platform. The matching process takes into account factors such as the partner's machining capacity, cost, and lead time to achieve efficient resource utilisation.

3.1.3. *Dynamic resource allocation.* The dynamic resource allocation module monitors the status of manufacturing resources in real time, including equipment usage, maintenance schedules and current load, to ensure dynamic and optimal allocation of resources. When the task matching is successful, the module dynamically adjusts the resource allocation scheme according to the urgency of the task and the immediate availability of the resources to avoid the over-occupation of bottleneck resources and ensure the smoothness of the production process.

3.1.4. *Production plan optimisation and execution.* The production plan optimisation component integrates advanced optimisation algorithms, such as improved hybrid metaheuristics, to optimise the processing sequence, equipment selection and operating time of production tasks to minimise the total production cycle time, taking into account the flexibility of the tasks and the dynamic changes in manufacturing resources. The optimised scheduling solution is subject to validation to ensure that all constraints are met including, but not limited to, lead time, cost and resource constraints. Once the solution has been validated, instructions will be issued to the production line to guide the actual production activities.

In summary, the cloud manufacturing production task scheduling framework builds a highly flexible and responsive production scheduling system by integrating the three functional modules of task decomposition and matching, dynamic resource allocation and production plan optimisation, which not only improves the production efficiency, but also effectively reduces the production cost and enhances the market competitiveness of the enterprise.

3.2. **Model description.** When exploring the production task scheduling problem under the cloud manufacturing model in depth, it is crucial to construct a rigorous mathematical model, which needs to fully reflect the diversity of production tasks and the flexibility of manufacturing resources. For the flexible production task scheduling problem, we further refine the model construction in order to guide the scheduling decision more accurately.

3.2.1. *Model variables and assumptions.* Let $J$ be the set of workpieces, where $J = \{J_1, J_2, ..., J_n\}$, $n$ are the number of workpieces, $M$ is the set of machines, where $M = \{M_1, M_2, ..., M_m\}$, $m$ are the number of machines, and each workpiece $J_i$ has $p_i$ process, which constitutes the set of processes $O_i = \{O_{i1}, O_{i2}, ..., O_{ip}\}$. Define $t_{ijk}$ as the processing time of the $j$ process of the workpiece $J_i$ on the machine $M_k$, $r_{ij}$ as the release time of the $j$ process of the workpiece $J_i$ (i.e., the completion time of the previous process), and $d_{ij}$ as the deadline for the $j$ process of the workpiece $J_i$. The main variables involved in the flexible production task scheduling problem model and their significance are shown in Table 1.

Table 1. Main variables involved in the model

| Notation | Connotation |
|---|---|
| $J$ | Workpiece set, $J = \{J_1, J_2, \ldots, J_n\}$, $n$ is the number of workpieces |
| $M$ | The set of machines, $M = \{M_1, M_2, \ldots, M_m\}$, $m$ is the number of machines |
| $O_i$ | The set of processes for the workpiece $J_i$, the $O_i = \{O_{i1}, O_{i2}, \ldots, O_{ip_i}\}$ |
| $p_i$ | Number of processes for workpiece $J_i$ |
| $t_{ijk}$ | Machining time of the first $j$ process of the workpiece $J_i$ on the machine $M_k$ |
| $r_{ij}$ | Release time of the $j$-th process of the workpiece $J_i$ |
| $d_{ij}$ | Deadline for completion of the first $j$ process of the workpiece $J_i$ |
| $C_{ij}$ | Completion time of the workpiece $J_i$ |
| $x_{ijk}$ | Decision variable, if the $j$-th process of the workpiece $J_i$ is processed on the machine $M_k$, then $x_{ijk} = 1$, otherwise $x_{ijk} = 0$ |

3.2.2. *Objective function.* The objective is to minimise the total completion time of all the workpieces (i.e., the last process completion time of the last workpiece) and the mathematical expression is shown as follows:

$$\text{Minimize} \quad Z = \max_{i \in J} \left\{ \max_{j \in O_i} \{C_{ij}\} \right\} \tag{3}$$

where $C_{ij}$ denotes the completion time of the workpiece $J_i$.

3.2.3. *Constraints.*

1. **Process machining sequence constraints:**
   For each workpiece $J_i$, the process order is fixed, i.e. if $O_{ij}$ must be completed before $O_{ik}$,
   $$\sum_{k=1}^{m} t_{ijk} + r_{ij} \leq \sum_{k=1}^{m} t_{ikj} + r_{ik}, \quad \text{for all } j < k \text{ and } i \in J \tag{4}$$
2. **Machine processing capacity constraints:**
   At the same moment, a machine can process only one process, i.e., if $O_{ij}$ and $O_{ef}$ are processed at the same time on machine $M_k$,
   $$\sum_{j=1}^{p_i} \sum_{k=1}^{m} x_{ijk} \leq 1, \quad \text{for all } i \in J, k \in M \tag{5}$$

3. **Time window constraints:**
Each process must start and finish within a given time window.

$$r_{ij} \leq \sum_{k=1}^{m} t_{ijk} x_{ijk} \leq d_{ij}, \quad \text{for all } i \in J, j \in O_i, k \in M \tag{6}$$

4. **Resource allocation constraints:**

Each machine can only be assigned to one process at any given time, ensuring that resources do not conflict.

$$\sum_{i=1}^{n} \sum_{j=1}^{p_i} x_{ijk} \leq 1, \quad \text{for all } k \in M \tag{7}$$

where $x_{ijk}$ is a decision variable indicating whether the $j$-th pass of the workpiece $J_i$ is performed on the machine $M_k$ or not, taking the value of 1 or 0.

Through the above model description, we not only define the mathematical framework of the problem, but also clarify the objectives and constraints of the solution, which provides a solid foundation for the subsequent algorithm design and optimisation.

4. **QoS constraint-based task scheduling model.** Before delving into the QoS constraint-based task scheduling model, we have constructed a hierarchical framework for hybrid task scheduling and proposed a mathematical model with the objective of minimising the total task operating time in Section 3.2. In order to further refine and ensure that the model satisfies users' QoS requirements, this section will focus on designing a task scheduling model that integrates QoS constraints to ensure that tasks are completed on time while also satisfying user-specific requirements on quality of service, such as response time, throughput, or resource utilisation.

4.1. **Definition of QoS constraints.** First, define the QoS constraints. For each task $T_i$, we define a set of QoS metrics set

$$Q_i = \{Q_{i1}, Q_{i2}, \ldots, Q_{ik}\},$$

where $Q_{ij}$ represents the $j$-th QoS metric for the $i$-th task, e.g., the maximum response time, the minimum throughput, or the maximum allowed delay. These metrics will be incorporated into the scheduling model as constraints to ensure that the scheduling scheme not only optimises the job time, but also meets the user-defined QoS criteria.

4.2. **Scheduling model construction.** Based on the above definitions, our goal is to construct a mathematical model that not only minimises the total operation time of all tasks, but also satisfies all QoS constraints. The model can be represented as:

$$\min Z = \sum_{i=1}^{n} t_i \tag{8}$$

where $Z$ denotes the total job time of all tasks and $t_i$ is the job time of task $T_i$.

At the same time, QoS constraints are introduced, e.g., for the maximum response time $Q_{\text{max\_res\_time}}$, which can be expressed as:

$$f_i \leq Q_{\text{max\_res\_time}}, \quad \forall i \in [1, n] \tag{9}$$

where $f_i$ is the completion time of the task $T_i$, which needs to meet the maximum response time specified by the user.

4.3. **Weighted multi-objective optimisation.** Considering that QoS metrics may conflict with each other, we can use a weighted approach to integrate multiple QoS metrics into a single objective function to balance different demands. Let the weight vector be $\boldsymbol{\omega} = (\omega_1, \omega_2, \ldots, \omega_k)$, the optimisation objective becomes:

$$\min Z = \sum_{j=1}^{n} \omega_j \cdot (Q_{ij} - Q_{\text{target\_}j})^2 + \sum_{i=1}^{n} t_i \tag{10}$$

where $(Q_{ij} - Q_{\text{target\_}j})^2$ measures the sum of squares of the deviation of each task from the target value on the $j$-th QoS metric, reflecting the degree of deviation from the desired QoS goal, and balancing time minimisation with QoS satisfaction by adjusting the weight $\omega_j$.

4.4. **Integration of resource constraints and QoS.** Resource constraints also need to be integrated in the model, e.g., to ensure that task scheduling does not exceed the resource limits of the partners. For each task $T_i$ to be executed on partner $P_k$, the resource constraint $R_{\text{limit}}$ needs to be satisfied:

$$\sum_{i \in R_k} r_{ik} \leq R_{\text{limit}}, \quad \forall k \in [1, m] \tag{11}$$

where $r_{ik}$ represents the amount of resources consumed by task $T_i$ on partner $P_k$.

5. **SFLA-based optimisation scheme design.** In this section, we will delve into the design of SFLA-based optimisation scheme for production task scheduling. The design aims to overcome the limitations of traditional scheduling algorithms, especially when facing highly dynamic and complex production tasks in cloud manufacturing environments, by effectively integrating the advantages of the Hadoop cloud computing manufacturing platform to ensure efficient and flexible task scheduling while satisfying QoS constraints.

5.1. **Initial population construction and fitness function.** First, we propose an innovative fusion strategy that combines the distributed processing capabilities of the Hadoop platform with the efficient search characteristics of SFLA. This fusion strategy aims to leverage Hadoop's MapReduce framework to process large-scale data, while exploiting SFLA's powerful global and local search capabilities to adapt to the multi-objective optimisation requirements in task scheduling. Specifically, we organically integrate the task decomposition and allocation mechanism in Hadoop with the population initialisation, local search and global update processes of SFLA. The initialisation population expression is shown as follow:

$$P_{\text{init}} = \{S_1, S_2, \ldots, S_n\} \tag{12}$$

where $P_{\text{init}}$ denotes the initial population, $S_i$ represents the $i$-th solution, and $n$ is the population size [26]. The construction of the population ensures that each solution has a certain initial quality to accelerate the convergence of the algorithm.

The fitness function is designed to evaluate the quality of the solution, combining the task completion time, resource consumption and QoS requirements. The specific form is given as follow:

$$f(S_i) = w_1 \cdot C(S_i) + w_2 \cdot T(S_i) - w_3 \cdot QoS(S_i) \tag{13}$$

where $C(S_i)$ is the total cost of completing the task; $T(S_i)$ is the total operating time of the task; $QoS(S_i)$ is the level of QoS satisfied; and $w_1$, $w_2$, and $w_3$ are trade-off factors.

5.2. **Introduction of adaptive movement factor.** In order to overcome the problem that the algorithm tends to fall into local optimum in local search, we design an Adaptive Movement Factor (AMF) to dynamically adjust the frog jump step size. The AMF varies with the number of iterations $t$ as follows:

$$AMF(t) = \frac{a \cdot \ln(t+1) + c}{t+b} \tag{14}$$

where $a$, $b$, $c$ are pre-set parameters; by adjusting these parameters, a larger search range can be maintained in the early stages of the algorithm, and progressively focusing on a finer search space as the iterations proceed to balance exploration and exploitation.

5.3. **Pheromone update mechanism.** In the SFLA algorithm, the pheromone update strategy is the key to guide the algorithm's search. We have adapted SFLA by combining the pheromone concept in ACO algorithm [27] and proposed a dynamic pheromone updating mechanism, which aims to improve the search efficiency of the algorithm and avoid premature convergence. The pheromone update rule is as follows:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho \cdot \sum_{k=1}^{K} \delta_{ik}(t) \cdot \phi(S_k) \tag{15}$$

where $\tau_{ij}$ denotes the pheromone concentration from node $i$ to node $j$, $\rho$ is the pheromone volatilisation rate, $\delta_{ik}(t)$ is an indicator function of the jump from node $i$ to node $j$ for the $k$-th individual at iteration $t$, and $\phi(S_k)$ is the fitness value of individual $S_k$.

Through the above design, our improved SFLA algorithm not only can effectively use the distributed computing capability of Hadoop cloud computing platform, but also enhances the search efficiency and global optimisation capability of the algorithm through the adaptive moving factor and dynamic pheromone updating mechanism, which ensures that the optimisation of the scheduling of the production tasks can be achieved under the premise of satisfying the QoS constraints.

## 6. Example analyses.

6.1. **Experimental environment and parameter settings.** In order to verify the effectiveness of the production task scheduling method based on QoS constraints and improved SFLA, we build a simulation experimental environment and set up the experimental parameters to ensure the reliability and comparability of the experimental results. The configuration of the experimental environment for this study is as follows:

**Hardware environment:** The experiment was conducted on a high-performance workstation equipped with Intel Core i7-10700K processor, 32GB RAM, NVIDIA GeForce RTX 3070 GPU, and the operating system is Windows 10 Pro 64-bit, which ensures that the algorithms run efficiently and stably.

**Software environment:** MATLAB R2023a was used as the main programming and simulation platform.

**Cloud platform simulation:** Considering the specificity of cloud manufacturing, the experiment simulates a simplified cloud manufacturing environment, using the CloudSim toolkit to simulate the allocation and management of cloud resources to ensure the closeness to real cloud manufacturing scenarios.

For the parameter settings, the following parameters were carefully chosen to ensure the effectiveness and usefulness of the algorithm: Problem size: 10 production scheduling problems in a representative mk series, including different numbers of workpieces ($n$ = 10–20), processes ($m$ = 5–15) and machines ($k$ = 4–15), were selected to comprehensively test the generalisation ability of the improved SFLA.

Population size: set to 100 to ensure that the algorithm is sufficiently exploratory in the search space while avoiding excessive computational burden.

Iteration number: the maximum number of iterations is set to 1000 for each group of experiments to ensure that the algorithm has enough time to converge to a better solution.

Algorithm parameters: to balance the exploratory and exploitative power of the algorithm, the pheromone volatility $\rho$ is 0.2, $a$ is 0.2, $b$ is 0.2, $c$ is 0.4, $w_1$ is 0.3, $w_2$ is 0.3, and $w_3$ is 0.5.

6.2. **Result analysis.** Under the same QoS constraints, PSO, original SFLA, and improved SFLA were used to solve the task scheduling model optimisation respectively, and the results are shown in Table 2. Where $n$ represents the number of workpieces to be processed, $m$ represents the number of available machines, and $w$ represents the total number of processes. $S$ is the optimal solution for 10 times, $Ave$ is the average value of the optimal solution for 10 times, and $T$ is the average time for 10 times for each algorithm.

Table 2. Task Scheduling Model Optimisation Solution

| **Issues** | $n \times m$ | $w$ | **PSO** | | | **Original SFLA** | | | **Improved SFLA** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $S$ | $Ave$ | $T$ | $S$ | $Ave$ | $T$ | $S$ | $Ave$ | $T$ |
| mk01 | $10 \times 6$ | 53 | 38 | 41.2 | 12.83 | 39 | 42.9 | 9.79 | 39 | 40.5 | 5.81 |
| mk02 | $10 \times 6$ | 58 | 29 | 30.0 | 14.71 | 24 | 29.8 | 11.66 | 25 | 26.1 | 7.68 |
| mk03 | $15 \times 8$ | 150 | 203 | 203.8 | 46.85 | 205 | 208.0 | 36.79 | 203 | 203.3 | 27.38 |
| mk04 | $15 \times 8$ | 90 | 60 | 62.7 | 30.71 | 61 | 66.6 | 20.64 | 61 | 61.5 | 17.40 |
| mk05 | $15 \times 4$ | 106 | 175 | 177.2 | 36.39 | 169 | 177.8 | 26.31 | 173 | 173.7 | 23.54 |
| mk06 | $10 \times 15$ | 150 | 58 | 59.4 | 62.52 | 59 | 62.2 | 52.43 | 58 | 61.0 | 41.12 |
| mk07 | $20 \times 5$ | 100 | 141 | 142.5 | 36.96 | 171 | 175.9 | 26.86 | 138 | 141.0 | 21.32 |
| mk08 | $20 \times 10$ | 223 | 525 | 527.4 | 65.89 | 527 | 531.7 | 55.78 | 322 | 527.1 | 47.04 |
| mk09 | $20 \times 10$ | 240 | 308 | 311.8 | 67.02 | 321 | 325.4 | 56.90 | 306 | 306.7 | 50.17 |
| mk10 | $20 \times 15$ | 240 | 227 | 231.1 | 69.38 | 231 | 240.0 | 59.25 | 213 | 215.5 | 52.94 |

It can be seen that the improved SFLA excels in finding the optimal solution in most scenarios. For example, in mk01 to mk05, the optimal solutions of the improved SFLA are close to those of the original SFLA, but in some complex scenarios such as mk07, mk08, and mk09, the optimal solutions of the improved SFLA are significantly better than those of the original SFLA and the PSO.This indicates that the improved SFLA is more effective in finding better solutions when solving task scheduling problems with larger scale or higher complexity . The improved SFLA provides better average optimal solutions in most cases. In particular,

in mk03 to mk10, the average optimal solution is not only better than the original SFLA, but also exceeds the PSO in many cases, showing the stability and reliability of the improved algorithm over multiple solving processes. This means that the improved SFLA can approach or reach the optimal solution more stably during multiple executions.

The improved SFLA achieved significant results in reducing the solution time. For all test cases, the average solution time of the improved SFLA is generally lower than that of the original SFLA and PSO, especially in mk01, mk02, mk05, and mk06, which show obvious time efficiency advantages. This indicates that the algorithm optimisation not only improves the solution quality, but also effectively reduces the consumption of computational resources and improves the practicality of the algorithm.

Taking the above analysis into account, the improved SFLA demonstrates superior performance in the optimal solution of task scheduling models. It reduces the solution

time while ensuring that a high-quality scheduling solution is found, which is especially important for cloud manufacturing environments that pursue both efficiency and accuracy. Improved SFLA enhances the algorithm's global and local search capabilities by optimising the skip-step

strategy, pheromone update mechanism, etc., which enables fast convergence to high-quality solutions while maintaining good computational efficiency when dealing with complex scheduling problems. Therefore, the improved SFLA is a powerful task scheduling tool for practical applications, especially in cloud manufacturing scenarios that satisfy stringent QoS requirements.

7. **Conclusion.** This study addresses the challenges of production task scheduling in Hadoop enterprise environments and proposes an innovative solution that combines QoS constraints with improved SFLA, aiming to improve the efficiency and quality of task scheduling. Through in-depth analysis of QoS multi-dimensional metrics, such as task response time, resource utilisation and system stability, our designed model is able to achieve efficient task scheduling in complex distributed computing environments, which ensures quality of service and optimal allocation of system resources. The experimental results and analyses confirm the following points:

(1) Importance of QoS constraints: Integrating QoS constraints into the scheduling process significantly improves the quality and real-time completion of critical tasks, proving that QoS constraints are a key factor in improving productivity and user satisfaction under the fulfilment of specific business requirements.

(2) Optimisation effect of SFLA: The improved SFLA demonstrates superior global search capability and fast convergence characteristics, and compared with other heuristic optimisation algorithms, it shows obvious advantages in reducing task waiting time and improving resource allocation balance, thus optimising the overall task execution efficiency.

(3) Advantages of the combined strategy: the proposed QoS-SFLA combined strategy maintains efficient scheduling while ensuring effective use of system resources and QoS guarantee for task execution, providing an ideal solution for balancing performance and resource constraints for massively parallel processing tasks on the Hadoop platform.

**Acknowledgment.** This work is supported by the Key Scientific Research Project of Hunan Provincial Department of Education: Research on Digital Transformation Technology and Path of Small and Medium-sized Enterprises (No. 23A0732), and the University Scientific And Technological Achievements Cultivation Project (No. 2021HYPY04).

## REFERENCES

[1] J. Wang and D. Li, "Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing," *Sensors*, vol. 19, no. 5, 1023, 2019.

[2] G. K. Manacher, "Production and stabilization of real-time task schedules," *Journal of The ACM (JACM)*, vol. 14, no. 3, pp. 439–465, 1967.

[3] R. Zhang, G. Li, T. Jiang, H. Chen, X. Li, W. Pei, and H. Xiao, "Incorporating production task scheduling in energy management of an industrial microgrid: A regret-based stochastic programming approach," *IEEE Transactions on Power Systems*, vol. 36, no. 3, pp. 2663–2673, 2020.

[4] T.-Y. Wu, H. Li, S. Kumari, and C.-M. Chen, "A Spectral Convolutional Neural Network Model Based on Adaptive Fick's Law for Hyperspectral Image Classification," *Computers, Materials & Continua*, vol. 79, no. 1, pp. 19–46, 2024.

[5] T.-Y. Wu, A. Shao, and J.-S. Pan, "CTOA: Toward a Chaotic-Based Tumbleweed Optimization Algorithm," *Mathematics*, vol. 11, no. 10, p. 2339, 2023.

[6] T.-Y. Wu, H. Li, and S.-C. Chu, "CPPE: An Improved Phasmatodea Population Evolution Algorithm with Chaotic Maps," *Mathematics*, vol. 11, no. 9, p. 1977, 2023.

[7] S. Bertsch, M. Schmidt, and P. Nyhuis, "Modeling of lateness distributions depending on the sequencing method with respect to productivity effects," *CIRP Annals*, vol. 63, no. 1, pp. 429–432, 2014.

[8] G. Patel, R. Mehta, and U. Bhoi, "Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing," *Procedia Computer Science*, vol. 57, pp. 545–553, 2015.

[9] I. Syed, "HAMM: A hybrid algorithm of Min-Min and Max-Min task scheduling algorithms in cloud computing," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 9, pp. 209–218, 2020.

[10] M. Iqbal, Z. Ullah, I. A. Khan, S. Aslam, H. Shaheer, M. Humayon, M. A. Salahuddin, and A. Mehmood, "Optimizing Task Execution: The Impact of Dynamic Time Quantum and Priorities on Round Robin Scheduling," *Future Internet*, vol. 15, no. 3, 104, 2023.

[11] S. S. Murad, R. Badeel, N. Salih, A. Alsandi, R. Faraj, A. Ahmed, A. Muhammed, M. Derahman, and N. Alsandi, "Optimized Min-Min task scheduling algorithm for scientific workflows in a cloud environment," *Journal of Theoretical and Applied Information Technology*, vol. 100, no. 2, pp. 480–506, 2022.

[12] S. Ishikawa, R. Kubota, and K. Horio, "Effective hierarchical optimization by a hierarchical multi-space competitive genetic algorithm for the flexible job-shop scheduling problem," *Expert Systems with Applications*, vol. 42, no. 24, pp. 9434–9440, 2015.

[13] S. A. Bello, L. O. Oyedele, O. O. Akinade, M. Bilal, J. M. D. Delgado, L. A. Akanbi, A. O. Ajayi, and H. A. Owolabi, "Cloud computing in construction industry: Use cases, benefits and challenges," *Automation in Construction*, vol. 122, 103441, 2021.

[14] B. Alouffi, M. Hasnain, A. Alharbi, W. Alosaimi, H. Alyami, and M. Ayaz, "A systematic literature review on cloud computing security: threats and mitigation strategies," *IEEE Access*, vol. 9, pp. 57792–57807, 2021.

[15] Z. Zhou, F. Li, H. Zhu, H. Xie, J. H. Abawajy, and M. U. Chowdhury, "An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments," *Neural Computing and Applications*, vol. 32, pp. 1531–1541, 2020.

[16] W. Li, J. Wu, Q. Zhang, K. Hu, and J. Li, "Trust-driven and QoS demand clustering analysis based cloud workflow scheduling strategies," *Cluster Computing*, vol. 17, pp. 1013–1030, 2014.

[17] U. Meyyappan, "Wavelet neural network–based wind speed forecasting and application of shuffled frog leap algorithm for economic dispatch with prohibited zones incorporating wind power," *Wind Engineering*, vol. 42, no. 1, pp. 3–15, 2018.

[18] W. Rogiest, K. Laevens, J. Walraevens, and H. Bruneel, "When random-order-of-service outperforms first-come-first-served," *Operations Research Letters*, vol. 43, no. 5, pp. 504–506, 2015.

[19] J. Xu and D. L. Parnas, "Priority scheduling versus pre-run-time scheduling," *Real-time Systems*, vol. 18, pp. 7–23, 2000.

[20] K. J. Merceedi and N. A. Sabry, "A comprehensive survey for hadoop distributed file system," *Asian Journal of Research in Computer Science*, vol. 11, no. 2, pp. 46–57, 2021.

[21] J. Zhang and M. Lin, "A comprehensive bibliometric analysis of Apache Hadoop from 2008 to 2020," *International Journal of Intelligent Computing and Cybernetics*, vol. 16, no. 1, pp. 99–120, 2023.

[22] N. Ahmed, A. L. Barczak, T. Susnjak, and M. A. Rashid, "A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench," *Journal of Big Data*, vol. 7, no. 1, 110, 2020.

[23] C.-S. Chiu and S. Ngo, "Hybrid SFLA MPPT design for multi-module partial shading photovoltaic energy systems," *International Journal of Electronics*, vol. 110, no. 1, pp. 199–220, 2023.

[24] P. Durgadevi and S. Srinivasan, "Resource allocation in cloud computing using SFLA and cuckoo search hybridization," *International Journal of Parallel Programming*, vol. 48, pp. 549–565, 2020.

[25] A. M. Rahmani, S. Ali, M. S. Yousefpoor, E. Yousefpoor, R. A. Naqvi, K. Siddique, and M. Hosseinzadeh, "An area coverage scheme based on fuzzy logic and shuffled frog-leaping algorithm (SFLA) in heterogeneous wireless sensor networks," *Mathematics*, vol. 9, no. 18, 2251, 2021.

[26] W. Ding, Y. Sun, L. Ren, H. Ju, Z. Feng, and M. Li, "Multiple lesions detection of fundus images based on convolution neural network algorithm with improved SFLA," *IEEE Access*, vol. 8, pp. 97618–97631, 2020.

[27] L. Wu, X. Huang, J. Cui, C. Liu, and W. Xiao, "Modified adaptive ant colony optimization algorithm and its application for solving path planning of mobile robot," *Expert Systems with Applications*, vol. 215, 119410, 2023.