Engineering Solutions to Mechanics, Marine Structures and Infrastructures

Eng. Solut. Mech. Mar. Struct. Infrastruct.
2024.1(3).7
ISSN：3006-2837
https://doi.org/10.58531/esmmsi/1/3/7

Article

Intelligent Scheduling System for Service Industry Based on Ant Colony Algorithm

Xuewen Chen[1,#], Chuantao Li[1,#], Yunyong Lu[1], Jianjun Zhu[1], Yize Tang[1], Ruihan Chen[1], Minhua Ye[2], Yueyang Wu[1], Yongpei Cao[1], Lele Xi[1], Jingxuan Cui[1], Hongrui Liu[1], Di Ning[3], Zhi Li[1,*]

[1] School of Mathematics and Computer, Guangdong Ocean University, 524088, Guangdong Zhanjiang China
[2] College of Ocean Engineering and Energy, Guangdong Ocean University, 524088, Guangdong Zhanjiang China
[3] College of Economics, Guangdong Ocean University, Zhanjiang 524088, China
[#] These authors contribute to this article equally.
[*] Corresponding Author: Zhi Li

Academic Editor: Dapeng Zhang < zhangdapeng@gdou.edu.cn>

Received: 24 July 2024; Revised: 20 August 2024; Accepted: 24 August 2024; Published: 26 August 2024

Abstract: Staff scheduling management is crucial in the service industry, especially in restaurants, supermarkets, and retail stores. Currently, shift preparation work mainly relies on manual operations, leading to inefficiency in the field of shift management in the service industry. Traditional monolithic scheduling systems face issues such as difficulty in horizontal expansion, tight coupling between modules, and challenges in updating and maintenance as functions increase, making them unsuitable for dynamic scheduling needs. In order to solve these problems, this study adopts the ant colony algorithm combined with TPE-BP neural network, web technology, and microservice architecture to develop an intelligent scheduling system. This system is divided into multiple service modules according to function, which effectively reduces the system coupling degree. It is highly scalable and can schedule shifts automatically, flexibly and efficiently, which improves the efficiency of schedule generation, and provides the service industry with a more convenient and intelligent scheduling management tool.

Keywords: Intelligent scheduling, Ant colony algorithm, Microservice architecture, Service industry

Citation: Chen X., Li Ch., Lu Y., Zhu J., Tang Y., Chen R., Ye M., Wu Yueyang 1, Cao Yo., Xi L., Cui J., Liu H., Ning D., Li Z. Intelligent Scheduling System for Service Industry Based on Ant Colony Algorithm. Eng. Solut. Mech. Mar. Struct. Infrastruct., 2024, 1(3), doi: 10.58531/esmmsi/1/3/7

1. Introduction

Staff scheduling management is of great significance in the service and retail industries. Currently, in the service industry, scheduling mainly relies on manual operations to allocate human resources and reduce costs. This involves manual calculations and creating schedules, making the process cumbersome and time-consuming, and leading to multiple issues. Furthermore, coordinating staff schedules across different stores must meet multiple constraints, including employee job matching and skill requirements, preferred working hours, and specific store needs [1]. Manual scheduling is prone to errors, often resulting in schedules that do not meet store or even industry requirements, making it difficult to ensure optimal scheduling outcomes. Additionally, the scheduling of shift workers is subject to numerous restrictions, such as continuous shift duration, rest periods, total weekly working hours, and total monthly working hours. These regulations impose constraints on employee scheduling in stores and directly impact employee compensation. However, manual scheduling struggles to ensure fairness and compliance with these regulations, leading to issues of inefficient human resource allocation [2].

Currently, researchers have been studying intelligent scheduling. In 1992, Melachrinoudis E. and Olafsson M. proposed a shift selection integer linear programming model, which was specifically applied to the Stop & Shop supermarket in Boston, Massachusetts at the time [3]. In 2018, Bing Song combined the greedy algorithm with web technology to develop an intelligent scheduling system for air traffic controllers, designed specifically for the Shenyang Air Traffic Control Center [4]. In 2023, Tianshu Xia et al. designed an intelligent scheduling algorithm for bank employees based on a genetic algorithm [5]. These studies have all improved the scheduling efficiency in their respective research areas, but their general applicability is limited. To meet the scheduling needs of service industry stores, it is crucial to develop an intelligent staff scheduling system that integrates automated schedule generation to replace traditional manual scheduling while providing convenient and secure scheduling data management.

Our contributions are summarized as follows:

- System architecture: Adopting B/S architecture with MVC model, Python is used for data processing and core algorithm development, and Java realizes server-side business logic. This architecture ensures complete decoupling of the user interface and server-side application logic, and adheres to the theory of distributed systems to improve the high availability and scalability of the system.
- .Demand forecasting: Introducing the TPE-BP neural network algorithm, it accurately predicts the future traffic flow based on the store's historical traffic data and other characteristics, providing a scientific basis for scheduling decisions.
- Intelligent scheduling optimization: Utilizing adaptive temperature-controlled ant colony algorithms, it automatically generates schedules that match employee work

preferences and store rules, dramatically increasing the level of scheduling automation.

- Scheduling management: The system supports administrators to generate, preview, save and query weekly schedules and provides detailed daily scheduling adjustment functions, which greatly enhances the flexibility and convenience of scheduling management.

2. System overall framework

Based on the modularization concept, this system is divided into three functional modules: the comprehensive information management module, the store demand forecasting module, and the intelligent employee scheduling management module.

After successfully logging into the system as an administrator, in the comprehensive information management module, the administrator has the authority to query store and employee information, and can perform operations such as adding, deleting, modifying, and querying information as needed. In the store demand forecasting module, the administrator can select specific time periods for predicting customer traffic, which helps analyze the number of employees required for the store. In the automated scheduling module, the administrator can obtain the number of employees needed for each time period and the corresponding schedule. Once the operations are completed, the schedule will be updated, and the administrator can safely log out of the system. An example of the main functional modules of this system is shown in Figure 1.

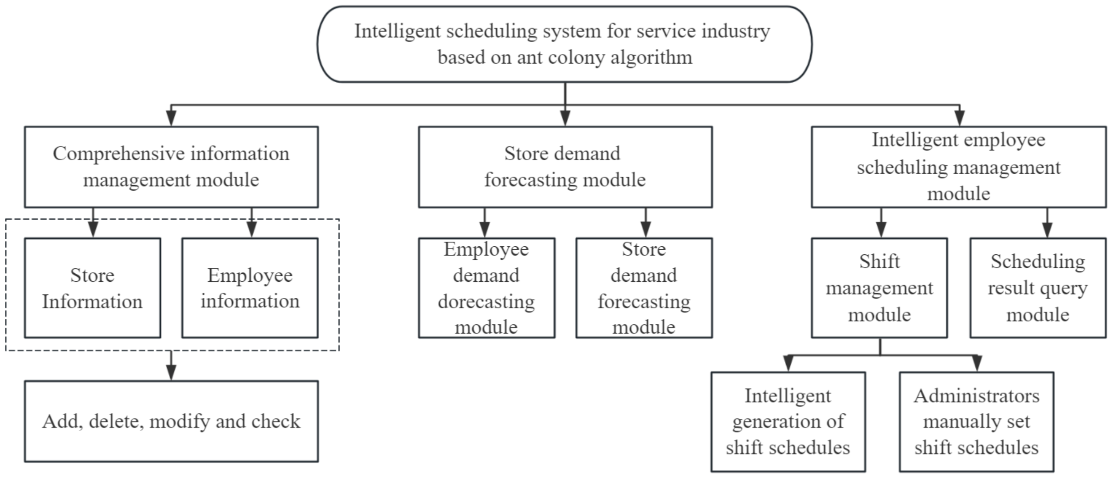
Intelligent scheduling system for service industry based on ant colony algorithm
Comprehensive information management module
Store demand forecasting module
Intelligent employee scheduling management module
Store Information
Employee information
Employee demand dorecasting module
Store demand forecasting module
Shift management module
Scheduling result query module
Add, delete, modify and check
Intelligent generation of shift schedules
Administrators manually set shift schedules

Figure. 1. Main functional modules of the intelligent scheduling system.

3. Database Design

The system uses a MySQL database. As a widely used open-source relational database management system, MySQL is suitable for applications of various data scales and is renowned for its exceptional performance, scalability, ease of use, security, and low cost. In this system, MySQL is primarily used to store store, employee, and scheduling-related information to achieve persistence of scheduling data. Details of the system's database tables are provided in Table 1.

Table 1. Database information table.

Table Name	Description	Information Included
t_ store	Store Information Table	Store name, store address, store area, number of employees, etc.
t_ employee	Employee Information Table	Employee name, gender, employee email, position, etc.
t_ employee_detail	Employee Detail Table	Employee age, phone number, home address, etc.
t_ employee_preference	Employee Preference Table	Preference type, preference value, notes, employee name, etc.
t_ schedule_info	Employee Scheduling Information Table	Scheduling date, store ID, employee ID, employee name, etc.
t_ scheduling_rule	Scheduling Rules Table	Scheduling rule type, store name, rule value
t_ shift_time	Fixed Shift Time Table	Store's fixed shift times

4. Store demand forecasting module algorithm flow

The store demand prediction module of this system employs the TPE algorithm to optimize the parameters of the BP neural network in order to achieve fully automatic prediction of store traffic. This approach achieves optimal traffic prediction without requiring the administrator to manually input the model parameters.

4.1 TPE Algorithm Principle

The TPE algorithm simultaneously models $p(x \mid y)$ and $p(y)$ instead of only modeling $p(y \mid x)$[6]. Here, $p(y)$ represents the distribution of the solutions, while $p(x \mid y)$ represents the distribution of the parameters x given the known solutions. The calculation of $p(x \mid y)$ is detailed in equation (1):

$$p(x \mid y) = \begin{cases} l(x) & y < y' \\ g(x) & y \geq y' \end{cases} \tag{1}$$

In the formula, y' is a predefined threshold; $l(x)$represents the density estimation of the observed value $x(i)$ with a loss function less than y'; $g(x)$represents the density estimation of the observed value $x(i)$ with a loss function greater than y'.

The definition of the optimization criterion EI is detailed in equation (2):

$$EI_{y^*}(x) \propto \left(y + \frac{g(x)}{l(x)}(1-y) \right)^{-1} \tag{2}$$

In the process of maximizing EI to find better hyperparameters, EI should gradually approach the sum of the highest probability $l(x)$ and the lowest probability $g(x)$.

4.2 Standard neural Network

A Standard neural network, also known as a BP neural network, is a type of multi-layer feedforward neural network structure, consisting of an input layer, hidden layers, and an output layer. The implementation steps of a Standard neural network are as follows:

Step 1: Input data sequence $N = (X_1, X_2 \cdots, X_m)$; map it to the hidden layer through weights and activation functions, generating a new sequence $\{W_1, W_2 \cdots, W_n\}$, which is the input value for the hidden layer. These values are computed through the activation function $\theta(\cdot)$ of the hidden layer, and then passed through the weights and activation function of the output layer to return the output values.

Step 2: Continuously adjust weights and thresholds through backpropagation of errors, with the derivation formulas detailed in equations (3) and (4):
Hidden layer weights:

$$\Delta w_{ij} = -\gamma \sum_{p=1}^{p} \sum_{n=1}^{N} \left(T_n^p - O_n^p\right) \cdot \phi'(net_n) \cdot w_{ji} \cdot \varphi'(net_i) \cdot x_j \tag{3}$$

Output layer weights:

$$\Delta w_{ij} = -\gamma \sum_{p=1}^{p} \sum_{n=1}^{N} \left(T_n^p - O_n^p\right) \cdot \varphi'(net_n) \cdot y_i \tag{4}$$

Where, i represents the hidden layer node, j represents the input layer node, k represents the output layer node, γ represents the learning rate, net_i represents the weighted sum of inputs to neuron i, w_{ij} represents the weight between node j and node i, n represents the input value of the jth node in the input layer, φ represents the activation function of the hidden layer, ϕ represents the activation function of the output layer, T_k^p represents the expected output value of the kth node in the output layer, and O_k^p represents the output value of the kth node in the output layer.

Step 3: The formula for calculating the model's mean square error is detailed in equation (5):

$$E_k = \frac{1}{2} \sum_{j=1}^{n} \left(Y_j^n - y_j^n\right)^2 \tag{5}$$

4.3 BP neural network based on TPE optimization parameters

In this system, the store demand forecasting module employs the TPE-BP neural network algorithm, which uses various factors as input features, including whether it is a holiday, the extent of promotional activities, peak working hours, weather conditions, and the average customer traffic during the same period in the past, while the customer traffic during this period of the current week is used as the output feature for training. The process of the TPE-BP neural network algorithm in this system is illustrated in Figure 2.

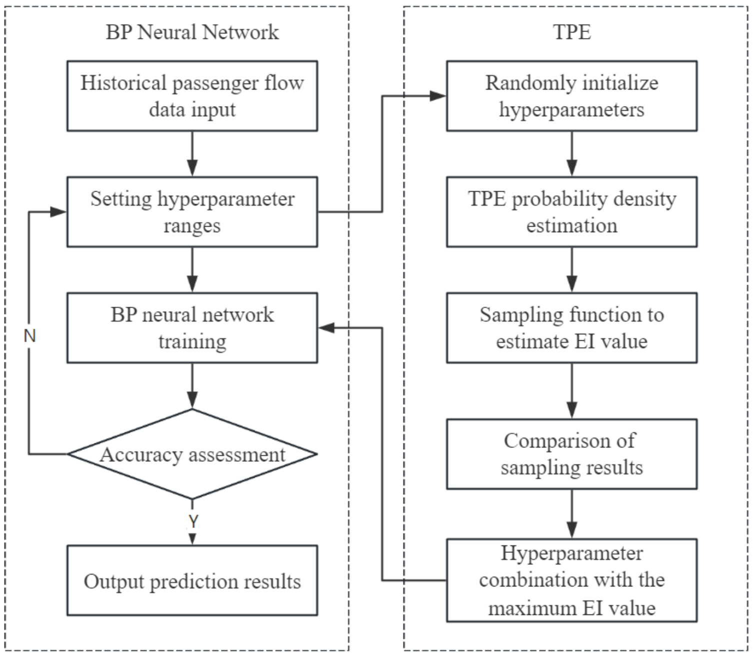
BP Neural Network
Historical passenger flow data input
Setting hyperparameter ranges
BP neural network training
Accuracy assessment
N
Y
Output prediction results
TPE
Randomly initialize hyperparameters
TPE probability density estimation
Sampling function to estimate EI value
Comparison of sampling results
Hyperparameter combination with the maximum EI value

Figure. 2. Flowchart of the TPE-BP neural network algorithm in this system.

5. Store demand forecasting module algorithm flow

5.1 Scheduling rules

In the case of manual scheduling, the scheduling rules are often diverse and lack standardization. However, when applying operations research algorithms to implement scheduling, these rules must be described in an expressible manner. As shown in Table 2, these rules need to be categorized and prioritized accordingly.

Table 2. Service store scheduling rules.

Rule Type	Rule Name	Description
Fixed Time	Business Hours	The time periods when the store is open and closed
	Opening Preparation	Duration required for preparation activities before opening
	Closing Cleanup	Duration required for cleanup activities after closing
Duration Limits	Shift Duration	Minimum and maximum working duration of a single shift
	Daily Working Hours	Minimum and maximum working hours per day for employees
	Weekly Working Hours	Minimum and maximum working hours per week for employees
	Rest Duration	Maximum duration of a single rest period
Staffing Configuration	Shift Staffing Needs	Number of employees required for normal shifts
	Minimum Staffing During No Traffic	Minimum number of staff required during no customer traffic
	Opening Staffing	Minimum number of staff required for opening preparation
	Closing Staffing	Minimum number of staff required for closing cleanup

	Skill Matching	Scheduling based on employees' skills
	Priority Assignment	Setting scheduling priority for employees
Leave Configuration	Leave Configuration	Setting employees' leave days for each week
	Special Holidays	Designating leave or advanced scheduling for special holidays
Position Restrictions	Prohibited Positions	Prohibiting employees with certain positions from being assigned to specific shifts

5.2 Ant Colony Algorithm

Ant Colony Optimization (ACO) is an optimization algorithm based on colony intelligence, which is inspired by the modeling of ants' foraging behavior. Ants release pheromone during the foraging process, which gradually evaporates over time. When an ant finds food, it leaves the pheromone on its path back to the nest. When other ants choose a path, they tend to choose the path with higher pheromone concentration, and gradually form an optimal path. This behavioral mechanism can be used to solve optimization problems.

In order to improve the performance of the algorithm, some control parameters are usually introduced, such as pheromone importance parameter, heuristic information importance parameter, pheromone volatilization coefficient and so on. The adjustment of these parameters can affect the convergence speed and the quality of the algorithm. In addition, the ACO algorithm can be combined with other optimization algorithms, such as local search algorithms, to further improve the solution efficiency and solution accuracy. Through these modeling and optimization processes, the ACO algorithm shows strong adaptability and high efficiency in solving combinatorial optimization problems such as traveler's problem, shortest path problem and scheduling problem.

Step 1: Algorithm Initialization: Set initial parameters, including the number of ants m, initial pheromone value t, pheromone importance parameter α, heuristic information importance parameter β, number of iterations, or termination conditions.

Step 2: Generate Initial Solution: Initialize the positions of all ants, with each ant randomly selecting a starting node. In each iteration, each ant moves through the graph according to a probability transition rule to construct a complete path.

Step 3: Probability Transition Rule: Ants decide their next move based on the pheromone concentration on the path and heuristic information. Paths with higher pheromone concentration and favorable heuristic information have a higher probability of being chosen, ensuring that the quality of the paths gradually becomes apparent during the search process.

Step 4: Calculate Path Length: After constructing a path, each ant calculates the path length (or total cost) to evaluate the quality of the path.

Step 5: Update Pheromones: Update the pheromones on the paths based on their quality. Pheromone updating involves two parts: first, increasing the pheromone on the path, with the increment related to the path's quality; second, pheromones on all paths evaporate at a certain rate to prevent premature convergence to a local optimum.

Step 6: Iteration and Termination Condition: Check whether the number of iterations has been reached or termination conditions are met. If not, return to Step 2 and continue iterating. If so, stop the algorithm and output the current best solution.

In summary, the Ant Colony Optimization algorithm simulates the process of ants searching for food in a graph. Through the positive feedback mechanism of pheromones, the pheromone concentration of good paths increases continuously, leading to convergence towards a global or near-optimal solution. With increasing iterations, the dynamic adjustments of pheromone evaporation and path quality evaluation enable the algorithm to effectively explore the solution space and avoid getting trapped in local optima.

6. Microservices Architecture Design

Microservices architecture is a distributed system architecture pattern that decomposes a large monolithic system into multiple relatively independent service modules based on functionality, facilitating development and maintenance, and utilizing containerization technology for service deployment. Compared to traditional monolithic architectures, microservices architecture features high scalability, high cohesion, low coupling, fault isolation, and efficient resource utilization, which enhance the system development and maintenance efficiency of development teams.

6.1 System Architecture Analysis

This system is well-suited for a microservices architecture, as it can be divided into multiple modules based on functionality, such as the comprehensive information management module and the intelligent employee scheduling management module, aligning with the microservices' service decomposition characteristic. Additionally, the scheduling system requires high scalability to handle the continuously changing scheduling workload of stores. Using appropriate technologies in implementing the comprehensive information management module and scheduling generation functionality can improve scheduling efficiency. The microservices architecture supports using different technologies for different services, making module development more flexible and independent.

When the system experiences high traffic for scheduling requests, a monolithic architecture might cause a single service to be overwhelmed by requests, potentially leading to system failure. However, adopting a microservices architecture allows for creating multiple scheduling microservice instances, utilizing a load balancer to distribute requests evenly across these instances. If a microservice instance encounters a fault, the load balancer will automatically isolate the faulty instance and redirect requests to other instances, ensuring high availability and reliability of the system. The request distribution process of the load balancer in this system is illustrated in Figure 3.

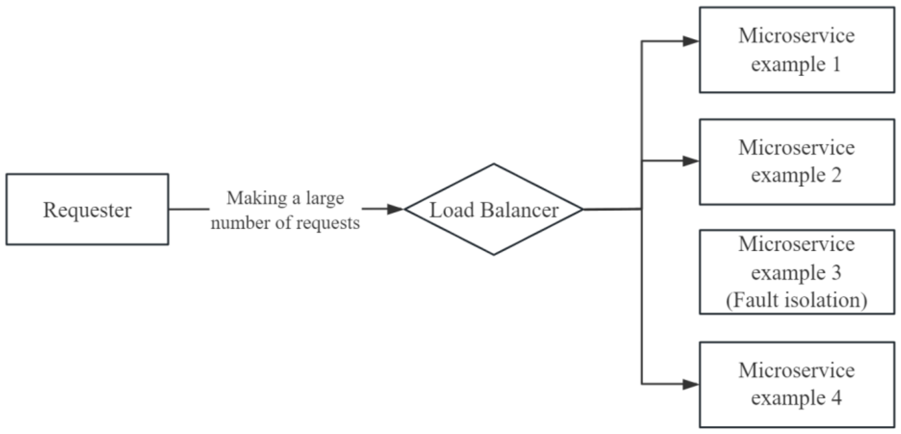

Figure. 3. Load balancer request distribution diagram.

6.2 Remote Invocation Between Services

In this system, service decoupling and deployment across different containers or hosts result in the need for remote invocation of services rather than local calls. To enable remote service interactions, the system utilizes OpenFeign technology, a RESTful API client based on the HTTP protocol with a declarative feature, commonly used with the Spring Cloud microservices framework. OpenFeign allows access to remote services via HTTP requests.

Integrating this technology involves simply adding dependencies, creating interface methods for the services to be called, and including necessary annotations to enable remote invocation. This approach offers good encapsulation, allowing callers to invoke methods based on the interface name without needing to understand the implementation details, similar to invoking local methods. The remote invocation process using OpenFeign is illustrated in Figure 4.

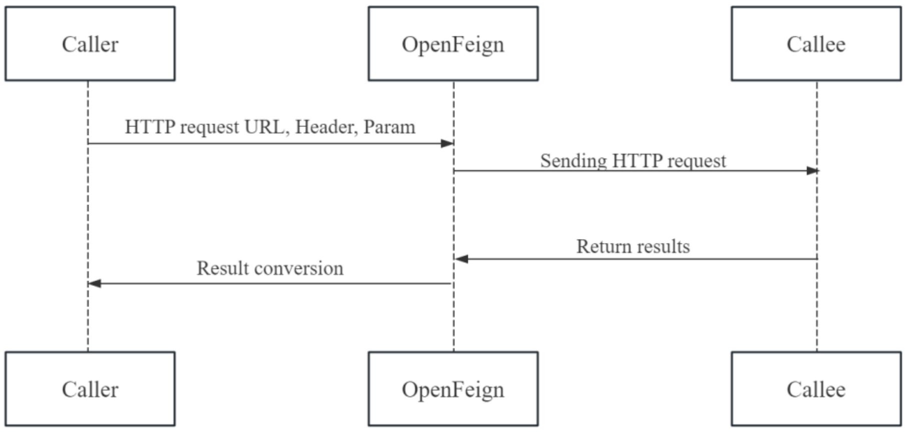

Figure. 3. OpenFeign remote call diagram in this system.

In the intelligent employee scheduling management module, the process for calling the comprehensive information management module to retrieve employee information and preferences is as follows. In this workflow, the intelligent employee scheduling management module acts as the consumer service, while the comprehensive information management module acts as the producer service.

1）Create a Feign Module: Add the OpenFeign dependency to the pom.xml project file. Include the necessary entity classes (e.g., Employee class, Employee Preference class) in the Feign module.
2）Create a Client Interface: In the Feign module, create an EmployeeClient interface class, using the @FeignClient annotation to specify the name of the producer service. Define interface methods ensuring that their paths and request methods align with the corresponding endpoints in the employee controller class.
3）Configure Feign Dependencies: Include the Feign module dependency in both the producer and consumer services, and configure Feign properties in their configuration files. Add the @EnableFeignClients annotation to the main classes of both services to enable Feign and specify the package path for scanning the Clients.
4）Implement Remote Call Interfaces: In the controller class of the producer service, write the necessary endpoints for remote calls, such as those for retrieving the employee list and employee preference list. In the service layer implementation class of the consumer service, inject an instance of the EmployeeClient class via dependency injection, and use the interface methods to obtain the return values.

7. Conclusions

This system uses employee scheduling in the service industry as a case study, optimizing permutations and combinations under certain conditions. The store demand forecasting module employs the TPE-BP neural network to fully utilize historical traffic data, achieving fully automated prediction of future store traffic. Additionally, the introduction of the Ant Colony Optimization algorithm into the intelligent scheduling module effectively addresses the issue of manually setting parameters in traditional ACO, enabling fully automated scheduling. The system features a user-friendly interface, easy operation, and high practicality, meeting the employee scheduling needs of service retail stores and providing a feasible solution for scheduling problems in the service industry.

In addition, integration of more advanced predictive models and optimization algorithms will be explored to address more complex scheduling needs. By continuously optimizing the user experience and system performance, we expect the system to become the industry standard for scheduling management in the service industry, providing more accurate and efficient solutions for all types of enterprises.

Acknowledgments: Thanks to each author for their contribution.

Funding: This project is funded by the Guangdong Provincial Science and Technology Innovation Strategic Special Fund (No. pdjh2023b0247); Guangdong Ocean University Undergraduate Innovation Team Project (No. CXTD2023014); Guangdong Ocean University Innovation and Entrepreneurship Training Program project (No. CXXL2024149).

Conflict of interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References:

1. Cheng DD, He LL. Retailer behavior feature mining based on genetic algorithm association rules. Industrial Control Computer. 2016; (8): 3.
2. Xia ZH, Pan WJ. Key technologies of intelligent scheduling system in call centers. Computer Engineering and Design. 2015; 36(5): 5. . https://doi.org/10.16208/j.issn1000-7024.2015.05.042
3. Olafsson M M .A scheduling system for supermarket cashiers[J].Computers & Industrial Engineering, 1992.DOI:10.1016/0360-8352(92)90078-X.
4. Song B. Design of an intelligent scheduling system for controllers. Information andComputer. 2018; (15): 3. DOI:CNKI:SUN:XXDL.0.2018-15-033..
5. Xia TS, Li YH, Xuan MH, et al. Design of an intelligent bank scheduling system based on genetic algorithm. Modern Information Technology. 2023; 7(12): 128-132.
6. Zha WT, Yan LC, Chen B, et al. Ultra-short-term regional wind power prediction based on TPE-LSTM. Computer Applications and Software. 2022; 39(11): 25-30.
7. Chen SZ, Zhang JH, Yu SS, et al. Wave temperature-controlled simulated annealing algorithm for solving the traveling salesman problem. Control and Decision. 2023; 38(4): 911-920.
8. Tang F, He YY. Solving 0-1 knapsack problem based on discrete binary particle swarm-simulated annealing algorithm. Industrial Control Computer. 2021. https://doi.org/10.3969/j.issn.1001-182X.2021.05.033
9. Wu B, Shi ZH. A segmented solution algorithm for TSP based on ant colony algorithm. Journal of Computer Research and Development. 2001; 24(12): 1328-1333. https://doi.org/10.3321/j.issn:0254-4164.2001.12.014