Robust Configuration Method Based on Multi-party Multi-objective Evolutionary Algorithm in Parallel Mutual Exclusion System

Jiaxin Zhou, Member, IAENG, Siyi Chen*, Haiyang Kuang, Xu Wang

***Abstract*—In the cloud computing environment, subsystems within the parallel mutual exclusion system compete for limited resources to maintain performance levels, particularly when faced with uncertain perturbations. This paper proposes a robust configuration method based on a multi-party multi-objective evolutionary algorithm to optimize server size and customer arrival rate, ensuring system performance remains at an acceptable level under perturbations. Initially, the minimum performance requirements of each subsystem are characterized in terms of waiting time, profit, and cost, and the corresponding feasible region is defined. Then, the concept of robustness is introduced into resource configuration to minimize the likelihood of performance degradation. The study treats the robust configuration of the parallel mutual exclusion system as a multi-party multi-objective problem and employs the latest multi-party multi-objective algorithm to compare with existing methods, determining the optimal configuration scheme. Experimental results demonstrate that the proposed configuration scheme not only enhances system robustness but also meets the differentiated requirements of each subsystem, significantly improving overall system performance and robustness.**

***Index Terms*—cloud computing, robustness, resource configuration, multi-party multi-objective optimization.**

I. Introduction

THE evolution of the Fifth Generation Mobile Communication System (5G) and the exponential surge in communication volume have positioned cloud computing technology as a key driving force for the advancement of information technology [1] [2]. Functioning as an innovative computing model, it has wielded significant influence across diverse realms such as science, industry, and social life. One fundamental advantage of cloud computing is its ability to provide highly flexible computing resources [3], thereby spurring the demand for large-scale data processing. Recognizing the potential of these computing resources, substantial attention has been directed towards the exploration of parallel systems, to augment task execution efficiency through parallel computing [4].

However, with the expansion of computing task scale and the increase in complexity, the perturbation issues faced by parallel systems in cloud computing environments are becoming increasingly significant [5]. When it comes to resource allocation, reasonable assumptions can be made about perturbations in the current cloud environment. On the one hand, in a situation of poor network conditions, servers may fail to receive client requests promptly, leading to timeouts. Alternatively, handling too many requests to exceed their capacities, the servers may also struggle to respond to customer requests promptly. On the other hand, in a situation of fewer processes to be handled [6], CPU cores cannot normally operate at full capacity and may experience faults after running for a sufficiently long time. If unexpected high loads occur, it will wake up dormant cores to provide more efficient services. We next analyze the results that may caused by the above two situations in detail. The former will lead to a decrease in the number of service requests completed per unit time, thereby reducing the revenue of cloud service providers and increasing the waiting time of customers. The latter will bring unnecessary waste of resources to increase the operating costs of the cloud platform [7]. These are contrary to the long-term operation of the cloud service system. Hence, from a long-term perspective, to fully protect the interests of cloud service providers and customers, the perturbation factor should be thoroughly taken into account. Considering this, the concept of robustness widely used in the field of control is introduced. Naturally, in the presence of perturbation, the problem of ensuring that the response characteristics of a cloud service system meet the required performance specifications can be regarded as a study of system robustness [8].

Based on the above analysis, the challenge we face is how to propose appropriate solutions for the robustness issues present in parallel systems [9]. In the field of cloud computing, optimizing resource utilization, reducing energy consumption, and minimizing response time are all benchmarks for effective task scheduling [10]. While covering the above elements, we hope to intuitively reflect the demands of those involved in cloud computing. Therefore the following three performance characteristics to assess the robustness of the system are constructed: the waiting time of customers, the revenue, and the cost of cloud service providers.

For a single system, the robustness problem involving multiple performance characteristics can usually be solved by a multi-objective evolutionary algorithm [11]. However, in parallel systems, multiple parallel tasks share limited

Manuscript received April 22, 2024; revised December 5, 2024. This work was supported in part by Excellent Young Scientists Fund of Hunan Provincial (Grant Nos. 22B0156) Basic and Applied Basic Research Foundation of Guangdong Provincial(Grant No. 2021A1515110383) and Key and Technology Projects of Foshan Shunde(Grant No.213021800254) .

Jiaxin Zhou is a postgraduate student at the School of Automation and Electronic Information, Xiangtan University, 411105, China (e-mail: jxzhou@smail.xtu.edu.cn).

Siyi Chen is an associate professor at the School of Automation and Electronic Information, Xiangtan University, 411105, China (corresponding author to provide e-mail: c.siyi@xtu.edu.cn).

Haiyang Kuang is a postgraduate student at the School of Automation and Electronic Information, Xiangtan University, 411105, China (e-mail: 202221623119@smail.xtu.edu.cn).

Xu Wang is a postdoctoral fellow in the Department of Industrial Engineering at Tsinghua University, 100084, China (e-mail: poorkongyiji@163.com).

resources. Since each subsystem needs to consider its response speed to meet the performance specifications, this leads to resource competition between subsystems [12]. Given this situation, we consider a parallel mutual exclusion system with conflicts of interest between subsystems. At the same time, the demands of cloud service providers and customers in the subsystem are not the same. For the complex scenario, the traditional configuration scheme based on a multi-objective evolutionary algorithm may not be suitable to provide a comprehensive and flexible solution. Considering this, we propose a robust configuration scheme based on a multi-party multi-objective algorithm according to the special demands of each subsystem. By rationally configuring the public resources that can be deployed in the parallel mutual exclusion system, our scheme ensures that each subsystem meets the robustness requirements to the greatest extent.

This paper studies the robust configuration scheme of parallel mutual exclusion systems. First, each subsystem is modeled as an M/M/m queuing system. Then, considering unpredictable disturbances, their impacts on the number of servers and the arrival rate of customers, in turn, effect the interests of cloud service providers and customers. Then, we construct the robust optimization problem of the system under disturbance conditions. On this basis,the optimal configuration scheme of the parallel mutual exclusion system is analyzed, which can improve the robustness of the system and optimize the waiting time, revenue and cost as much as possible. The main contributions of this paper are summarized as follows.

- Consider a parallel mutual exclusion multi-server system to depict a scenario where subsystems in cloud services compete for limited system resources.
- Design the boundedness constraints for the variations of servers size and arrival rate of customers impacted by unpredictable perturbations in cloud computing system, which reflects the worst cases that cloud service providers and customers can accept.
- Propose a robust analysis method based on multi-objective algorithm to improve the robustness of the system as much as possible.
- Conduct a series of experiments to prove the effectiveness of our proposed method in solving such problems.

The structure of this paper is as follows: Section II provides a review of related work. Section III builds the system framework as well as the revenue model and cost model. Section IV clarifies the optimization problem and robustness metric. Section V proposes an optimal configuration scheme for the robustness problem in parallel mutual exclusion systems. Section VI concludes this work.

II. Related Work

In this section, we first review recent work related to queuing system modeling problems. Subsequently, we explore relevant research on resource allocation issues in the field of cloud computing. Finally, we provide a detailed review of various literature about task scheduling in the realm of cloud computing.

In the field of cloud computing, the discussion on queuing models has been a persistent focal point. Evaluating and refining queuing models is crucial to ensuring the seamless operation of cloud services and enhancing the quality of service delivered to customers. A. Gorbunova et al. [13] considered a fork-join queuing system with Pareto-distributed service times on the servers. The average response time and its standard deviation were analyzed using a novel approach that combines simulation modeling with machine learning methods. Wang et al. [14] proposed service performance analysis models based on Geo/G/1 queuing systems and server vacation queuing systems. In these models, various parameters related to key performance indicators of cloud computing servers and optimization issues of cloud computing concurrency were discussed. Chen et al. [15] analyzed the profit maximization problem with deadline constraints in a loop-cascaded queuing system. They proposed a heuristic algorithm to search for high-quality solutions, aiming to achieve the optimal configuration of a cloud computing platform. Lv et al. [16] analyzed the effect of server breakdowns and negative customers on the queuing system.

After completing the modeling work of the cloud service system, many articles focus on obtaining an efficient resource allocation scheme to improve system performance. S. Souravlas et al. [17] proposed a fair resource allocation scheme with flow control and maximum utilization of the systems resources. Jia et al. [18] designed and implemented an efficient memory allocation scheme and analyzed the impact of workload memory intensity on system performance. K. Karthiban et al. [19] proposed a green computing fair resource allocation model utilizing deep reinforcement learning, providing an efficient resource allocation solution for customers in the network. Yuan et al. [20] designed a profit-maximizing collaborative computation offloading and resource allocation algorithm to maximize the system's profit while ensuring strict adherence to task response time constraints.

In addition to the rational allocation of deployable resources within the system, task scheduling is also an essential part of cloud computing to achieve optimal resource utilization, reduce energy consumption (EC), minimum response time, and maximum efficiency [21]. R. Stewart et al. [22] proposed the first Pareto-efficient task scheduling method based on MIP (Mixed Integer Programming) formulations to minimize both makespan and energy consumption. Abdelmoneem et al. [23] presented a heuristic-based scheduling algorithm considering perceptual mobility to minimize the time and cost of task completion.

III. MODEL DESCRIPTION

A. Multi-server system

In this paper, we study a multi-server system with multiple subsystems $S_1, S_2, ...S_n$ arranged in a parallel structure in cloud computing, which is shown in Figure.1. The notations used are listed in Table I. The system is described in detail by the following points. First, we use the M/M/m queuing model to model the subsystems, and each subsystem can handle the requests of customers separately. Second, requests from customers to the cloud service system are assigned to the subsystems for service. Third, a certain number of servers in the system will also be assigned to the subsystems to process those requests. Hence, it is not difficult to find the resource competition relationship among the subsystems.

TABLE I: Notations used in this paper.

Symbol	Definition	Symbol	Definition
S_i	Subsystem i	m	The number of servers (Server size)
λ	The number of arrival customer per (unit time) UT	s	Server speed
m_{all}	The whole server size	m_i	Server size in S_i
λ_{all}	The whole number of arrival customer per UT	λ_i	The number of arrival customer in S_i
$\bar{j}$	Average task execution demand	μ	The number of completed service per UT
ρ_i	The utilization factor of servers in S_i	P_{k_i}	The probability that k service requests in S_i
P_{q_i}	The probability to wait in S_i	W_i	Waiting time of customers in S_i
$f_{w_i}(t)$	The pdf of waiting time	$u(t)$	Unit impulse function
T_i	The expectation of waiting time in S_i	R_i	Revenue in S_i
C_i	Cost in S_i	a	Fees per one billion instructions
D_L	Maximum tolerable time that customers can wait	β	Rental price of a server per UT
N_{sw}	Average gate switching factor per clock cycle	C_l	Load capacitance
V	Supply voltage	f	Lock frequency
ϕ	A positive constant	b	A positive proportional coefficient
P_d	Dynamic power consumption	P^*	Static power consumption
δ	Electricity price of energy	H^*	A $working\ point$ in $feasibel\ region$
H	A point on boundary	$Dist_i$	The distance between H^* and H
r^*	Shortest robustness radius	θ	Polar angle in vector form
$\mathbb{1}_1$	Indicator function 1	$\mathbb{1}_2$	Indicator function 2
D_i	The lowest demand on waiting time in S_i	r_1^*	Robustness radius to the lowest demand on waiting time
r_2^*	Robustness radius to the lowest demand on revenue	r_3^*	Robustness radius to the lowest demand on cost

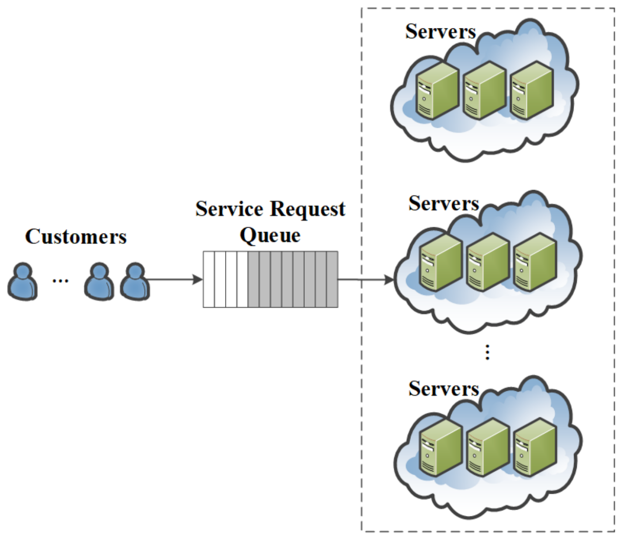

Fig. 1: Multi-server queuing system.

Without loss of generality, we merely analyze the multi-server system with two subsystems S_1, S_2, i.e., $n = 2$. The whole number of servers in the system is donated as m_{all}, and all the servers run at the same running speed s. Among them, we assume that the number of servers assigned to subsystem S_i is m_i (i=1,2). Then, we have the equation $m_2 = m_{all} - m_1$. Considering the randomness of customer's arrival, we assume it follows a Poisson process, where λ_{all} represents customer arrival rate, namely, the average number of customers arriving at the system per unit time. Subsequently, let λ_i (i=1,2) represent the number of service requests assigned to subsystem S_i. Then, we have the equation $\lambda_2 = \lambda_{all} - \lambda_1$. Next, we consider the number of instructions contained by each service request to measure its task execution demand, which is denoted by an exponential random variable j with mean $\bar{j}$. Thus, the execution time of a task on the multi-server system can also be considered as an exponential random variable $t = j/s$ with mean $\bar{t} = \bar{j}/s$. Moreover, the service rate can be denoted as $\mu = 1/\bar{t} = s/\bar{j}$ and the utilization factor in subsystem S_i is defined as $\rho_i = \lambda_i/m_i\mu = \lambda_i/m_i \times \bar{j}/s$, which means the percentage of the average time when the server is in busy state. Let p_{k_i} denote the probability that there are k_i service requests (waiting or being processed) in subsystem S_i. Then, we have

$$p_{k_i} = \begin{cases} p_{0,i}\frac{(m_i\rho_i)^k}{k_i!}, & k_i < m_i \\ p_{0,i}\frac{{m_i}^{m_i}\rho_i^k}{m_i!}, & k_i \geq m_i \end{cases} \tag{1}$$

where

$$p_{0,i} = \left(\sum_{k_i=0}^{m_i-1} \frac{(m_i\rho_i)_i^k}{k_i!} + \frac{(m_i\rho_i)^{m_i}}{m_i!}\frac{1}{1-\rho_i} \right)^{-1} \tag{2}$$

To ensure the ergodicity of the queue system, it is obvious that the condition $0 < \rho_i < 1$ should be satisfied.

On this basis, when all servers in a multi-server system are busy, newly arrived service requests have to wait in the queue. We express the probability of such a case as follows.

$$p_{q_i} = \sum_{k_i=m_i}^{\infty} p_{k_i} = \frac{p_{m_i}}{1-\rho_i} = p_{0,i}\frac{(m_i\rho_i)^{m_i}}{m_i!} \cdot \frac{1}{1-\rho_i} \tag{3}$$

B. Waiting time distribution

Generally, customers hope to get the executions of their service requests as soon as possible, otherwise, they are likely to leave if they wait longer than they can tolerate. Hence, cloud service platforms have to keep the waiting time of customers from exceeding the tolerable deadline constraint. Considering this, the modeling work of the waiting time of customers is necessary to be carried out for the numerical analysis. With this premise, we denote W_i as the waiting time of the service request which is handled in subsystem S_i. Notice that, the probability distribution function (pdf) of the waiting time in the M/M/m queuing system has already been fully considered in previous literature, which is shown as follows [24][25].

$$f_{W_i}(t) = (1 - p_{q_i})\, u(t) + m_i \mu p_{m_i} e^{-(1-\rho_i) m_i \mu t} \tag{4}$$

where $p_{m_i} = p_{0,i}(m_i \rho_i)^{m_i}/m_i!$ and $u(t)$ is unit impulse function, which is defined as

$$u_z(t) = \left\{ \begin{array}{ll} z, & 0 \le t \le \frac{1}{z} \\ 0, & t > \frac{1}{z} \end{array} \right. \tag{5}$$

Let $z \to \infty$, then we have

$$u(t) = \lim_{z \to \infty} u_z(t) \tag{6}$$

The function $u_z(t)$ has the following properties

$$\int_0^\infty u_z(t)\, dt = 1 \tag{7}$$

and

$$\int_0^\infty u_z(t)\, dt = z \int_0^{1/z} t dt = \frac{1}{2z} \tag{8}$$

Further, according to (4) and (7), we can calculate the expectation of waiting time as follow.

$$\begin{aligned} T_i &= \int\limits_0^{+\infty} t \cdot f_{W_i}(t)\, dt \\ &= \int\limits_0^{+\infty} t \cdot \left((1 - p_q)\, u(t) + m_i \mu p_m e^{-(1-\rho_i) m_i \mu t} \right) dt \\ &= \int\limits_0^{+\infty} t \cdot m_i \mu p_m e^{-(1-\rho) m_i \mu t} dt \\ &= \frac{p_m}{m_i \mu (1-\rho_i)^2} \end{aligned} \tag{9}$$

C. Revenue modeling

Generally, cloud services are not provided for free, so customers whose requests are handled have to pay for the services. To evaluate the relationship between the quality of services and the corresponding charges to the customers, a Service Level Agreement (SLA) is adopted. In this paper, we use flat rate pricing for service requests because it is relatively intuitive and easy to obtain. Based on this, the service charge function Z can be expressed as

$$Z(rj, W_i) = \left\{ \begin{array}{ll} aj, & 0 \le W_i \le D_L \\ 0, & W_i > D_L \end{array} \right. \tag{10}$$

where a is a constant that represents the fee per unit of service, and D_L is the maximum tolerable time that the customer can wait (i.e., the deadline). In this paper, when the waiting time does not exceed the deadline, the fee that the customer needs to pay is considered as a constant value. Otherwise, the customer will enjoy the services for free. Based on (4) and (10), we can calculate the expectations of $Z(j, W)$

$$\begin{aligned} Z(j) &= H\left(Z\left(j, W_i\right)\right) \\ &= \int_0^\infty f_{W_i}(t) aj dt \\ &= aj \int_0^\infty \left[(1 - P_q)\, u(t) + m_i \mu p_m e^{-(1-\rho_i) m_i \mu t} \right] dt \\ &= aj \int_0^{D_l} \left[(1 - P_q)\, u(t) + m_i \mu p_m e^{-(1-\rho_i) m_i \mu t} \right] dt \\ &= aj \left[(1 - P_q) - \frac{p_m}{1-\rho_i} \left(e^{-(1-\rho_i) m_i \mu D_L} - 1 \right) \right] \\ &= aj \left(1 - \frac{p_m}{1-\rho_i} e^{-(1-\rho_i) m_i \mu D_L} \right) \end{aligned} \tag{11}$$

Notice that, the task execution requirement j is also a random variable, which follows the exponential distribution. On this basis, the expected charge of a service request in the multi-server system can be calculated as follows.

$$\begin{aligned} \bar{Z} &= H\left(Z(j)\right) \\ &= \int_0^\infty \frac{1}{\bar{j}} e^{-z/\bar{j}} Z(z) dz \\ &= \frac{a}{\bar{j}} \left(1 - \frac{p_{m_1}}{1-\rho_i} e^{-(1-\rho_i) m_i \mu D_L} \right) \int_0^\infty e^{-z/\bar{j}} z dz \\ &= a\bar{j} \left(1 - \frac{p_m}{1-\rho_i} e^{-(1-\rho_i) m_i \mu D_L} \right) \end{aligned} \tag{12}$$

Further, we can also obtain

$$F_{W_i}(D_L) = 1 - \frac{p_m}{1-\rho_i} e^{-(1-\rho_i) m_i \mu D_L} \tag{13}$$

Since the number of service requests assigned to subsystem S_i is λ_i (hundred service requests per hour), the total revenue of S_i is $\lambda a\bar{j}$ if all the service requests could be served before the deadline. However, if parts of the service requests are served for free, the actual revenue that the cloud service providers earn from S_i can be described as

$$R_i = \lambda_i F_{W_i}(D_L)\, a\bar{j} \tag{14}$$

D. Cost modeling

The costs to the cloud service provider consist of two main components, namely the cost of infrastructure leasing and energy consumption. The infrastructure provider maintains a large number of servers for loan, and the cloud service provider rents the servers on request and pays the corresponding leasing fees. Assume that the rental price of one server per unit of time is β, then the server rental price of the multi-server system with m_{all} servers is $m_{all}\beta$ in total.

The cost of energy consumption depends on the price of electricity and the amount of energy consumed. In this paper, the following dynamic power model is used to express energy consumption, which has been discussed in many literature [26].

$$P_d = N_{sw} C_L V^2 f \tag{15}$$

where N_{sw} is the average gate switching factor per clock cycle, C_L is the load capacitance, V is the supply voltage, and f is the clock frequency.

In the ideal case, the relationship between the supply voltage V and the clock frequency f can be described as $V \propto f^{\phi}(0 < \phi \leq 1)$. The execution server speed s is usually linearly related to the clock frequency, namely, $s \propto f$. Therefore, the dynamic power model can be converted to $P_d \propto bN_{sw}C_L s^{2\phi+1}$. For the sake of simplicity, we can assume $P_d = bN_{sw}C_L s^{2\phi+1} = \xi s^{\alpha}$, where $\xi = bN_{sw}C_L$ and $\alpha = 2\phi + 1$. In this paper, we set $N_{sw}C_L = 4$, $b = 0.5$, ϕ=0.55. Hence, α=2.1 and ξ=9.4192. In addition to dynamic power consumption, each server also consumes a certain amount of static power consumption P^*.

Due to the server utilization ρ_i affecting the dynamic power consumption of the server, the average amount of energy consumption per unit of time is $P = \rho_i \xi s^{\alpha} + P^*$. Assuming an electricity price of δ per watt, the total cost per unit time for the cloud service provider to maintain subsystem S_i can be described as

$$C_i = m_i(\beta + \delta(\rho_i \xi s^{\alpha} + P^*)) \tag{16}$$

IV. Problem description

According to the descriptions in the previous section, the interests concerned by customers and cloud service providers are different. In terms of optimization objectives, maximizing the demands of customers or cloud service providers is considered an optimization objective by most studies. However, they ignore the existence of perturbation factors in real cloud environments. Affected by those perturbation factors, the decline of system performance makes the optimal results can be hardly maintained. In this paper, we consider the impact of perturbation factors that may lead to a decrease in revenue and an increase in cost for cloud service providers, or an increase in the waiting time of customers. Then, a distance metric scheme is designed, which can numerically measure the ability of the system to resist performance reduction caused by perturbation factors.

A. Robustness analysis

According to (9) and (14), because of the sophisticated structure of p_m, the waiting time of customers and the revenue of cloud service providers are difficult to analyze numerically. For the sake of simplicity, the following approximations are adopted, i.e., $\sum_{k=0}^{m_i-1} \frac{(m_i\rho_i)^k}{k!} \approx e^{m_i\rho_i}$ and $m_i! \approx \sqrt{2\pi m_i}\left(\frac{m_i}{e}\right)^{m_i}$ [27]. On this basis, p_m can be rewritten as follows.

$$p_m \approx \frac{1-\rho_i}{\sqrt{2\pi m_i}\,(1-\rho_i)\left(e^{\rho_i}/e\rho_i\right)^{m_i} + 1} \tag{17}$$

Then, substituting (17) into (13), we have.

$$F_{W_i}(D_L) \approx 1 - \frac{e^{-m_i\mu(1-\rho_i)D_L}}{\sqrt{2\pi m_i}\,(1-\rho_i)\left(e^{\rho_i}/e\rho_i\right)^{m_i} + 1} \tag{18}$$

Further, according to (17) and (18), we can obtain the approximations of the waiting time of customers as well as the revenue of cloud service providers as follows.

$$T_i = \frac{(\lambda_i e)^{m_i}}{(sm_i)^{m_i-1}(sm_i-\lambda_i)^2 e^{\frac{\lambda_i}{s}}\sqrt{2\pi m_i} + (sm_i-\lambda_i)\,(e\lambda_i)^{m_i}} \tag{19}$$

and,

$$R_i = \lambda_i a\bar{r}\left[1 - \frac{e^{-m_i\mu_i(1-\rho_i)D_L}}{\sqrt{2\pi m_i}(1-\rho_i)\left(e^{\rho_i}/e\rho_i\right)^{m_i} + 1}\right] \tag{20}$$

In this paper, we consider the perturbation factors as the unpredictable variations in the server size and the arrival rate of customers. Notice that, the utilization factor ρ_i in (16)(20) can be expressed by $\rho_i = \frac{\lambda_i \bar{r}}{m_i s}$. To describe the impact of the server size and the arrival rate of customers on the revenue and the cost of cloud service providers more clearly, by substituting the expression of ρ_i into (20), and then setting $\bar{r}$ to 1 hundred billion instructions, we have.

$$R_i = \lambda_i a\left[1 - \frac{e^{(-sm_i+\lambda_i)D_L}(e\lambda_i)^{m_i}}{\sqrt{2\pi m_i}(sm_i-\lambda_i)e^{\lambda_i/s}(sm_i)^{m_i+1} + (e\lambda_i)^{m_i}}\right] \tag{21}$$

and,

$$C_i = \left(m_i\beta + \delta\lambda_i\xi s^{\alpha-1} + m_i\delta P^*\right) \tag{22}$$

Considering (19) (21) and (22), we can express these formulas by $T_i = f(m_i, \lambda_i)$, $R_i = r(m_i, \lambda_i)$ and $C_i = c(m_i, \lambda_i)$ respectively. Obviously, (19) (21) and (22) show the strong nonlinear characteristics, which will result in the difficulty in numerical analysis. In this case, we choose to draw the graphics of $T_i = f(m_i, \lambda_i)$, $R_i = r(m_i, \lambda_i)$ and $C_i = c(m_i, \lambda_i)$ to show the functional relationship intuitively.

By setting $s = 2$ hundred billion instructions per hour, $a = 10$ dollars per one hundred billion instructions (Note: The monetary unit dollars in this paper may not be identical but should be linearly proportional to the real US dollars.), $\beta = 1.5$ dollars per one hundred billion instructions, $\delta = 0.1$ dollars per watt, $P^* = 2$ watts (Note: These parameters are chosen only for illustration and should be scaled to any values.), and then traversing the size and arrival rate of the servers in $[4, 20]$ and $[4, 14]$, we draw the three-dimensional surfaces of (19) (21) and (22), which are shown in Figure.2(a) Figure.2(b) and Figure.2(c) respectively. From Figure.2(a), it can be observed that as the server size decreases and the arrival rate increases, customers have to spend longer waiting times to receive service, and vice versa. While for Figure.2(b), there are cases where the server size is large but the arrival rate is low, meaning the powerful service capabilities are far beyond the customer's demands. This results in a few parts of the service resources being used and brings a little revenue to cloud service providers while most of them are wasted. Then, when the server size remains constant, the increase in the arrival rate of customers reduces the waste of resources, generating higher revenue for cloud service providers. However, if the server's size decreases while the arrival rate of the customers is excessively high, the provider's revenue may decrease. This is because the service capacity becomes insufficient to fulfill an adequate number of requests before the deadline. Consequently, some customers may enjoy services without charge due to the long wait. From Figure.2(c), it can be observed that as the server's size and arrival rate increase, the costs incurred by cloud service providers will continue to rise, and vice versa.

Considering the waiting time of customer to be a deterministic value, i.e., $f(m_i, \lambda_i) = b_1$ (b_1 is a constant), we can

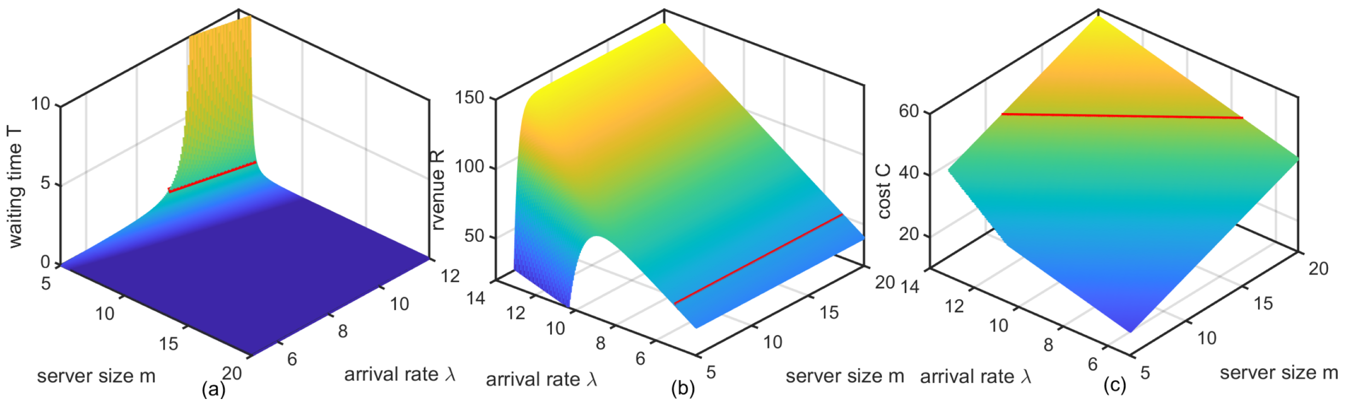

Fig. 2: (a)The mesh of waiting time T_i versus m_i and λ_i. (b)The mesh of revenue R_i versus m_i and λ_i. (c)The mesh of cost C_i versus m_i and λ_i.

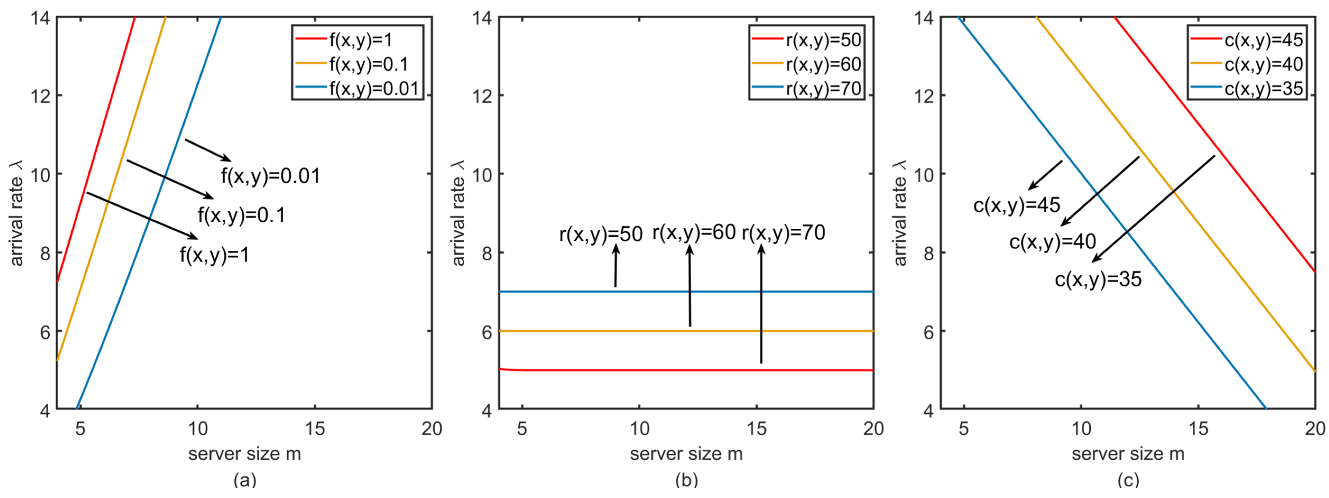

Fig. 3: (a)Deadline D_i versus m_i and λ_i. (b)Revenue R_i versus m_i and λ_i. (c)Cost C_i versus m_i and λ_i.

find the implicit function relationship between the servers size and arrival rate in (19), which can be plotted as the red curve in Figure.2(a). By specifying $b_1 \in [0.01, 0.10, 1.00]$, and mapping the implicit function relationship into (m_i, λ_i) plane, three curves can be drawn in Figure.3(a). On this basis, when we set the deadline as $D_i = 0.1$, only if the server size and the arrival rate are selected in the region below the curve bottom in Figure.3(a) can the customer requests be served before the deadline. In other words, such region can be named as the $feasible\ region$ with the waiting time of customers being taken into consideration.

Similarly, considering the revenue of cloud service provider to be a deterministic value, i.e., $r(m_i, \lambda_i) = b_2$ (b_2 is a constant), we can find the implicit function relationship between the servers size and arrival rate in (21), which can be plotted as the red curve in Figure.2(b). By specifying $b_2 \in [50, 60, 70]$, and mapping the implicit function relationship into (m_i, λ_i) plane, three curves can be drawn in Figure.3(b). On this basis, when we set the acceptable lowest revenue as $R_i = 60$, only if the server size and the arrival rate are selected in the region above the curve top in Figure.3(b) can the cloud service providers gain more revenue. In other words, such a region can be named as the $feasible\ region$ with the revenue of cloud service providers being taken into consideration.

Furthermore, we consider the definite value of the cloud service provider's costs in Figure.2(c), where $c(m_i, \lambda_i) = b_3$, and b_3 is a constant. We can still observe the relationship between server size and arrival rate as depicted by the red line in Figure.3(c) based on (22). By specifying $b_3 \in [35, 40, 45]$, and mapping the implicit function relationship into (m_i, λ_i) plane, three curves can be drawn in Figure.3(c). On this basis, when we set the acceptable maximum costs as $C_i = 40$, only if the server size and the arrival rate are selected in the region below the curve bottom in Figure.3(c) can the cloud service providers spend less cost. Similarly, considering the costs of the cloud service provider, this region can be named the $feasible\ region$.

Based on the above discussion, the impacts of the perturbation factors on the revenue and the cost of cloud service providers as well as the waiting time of customers can be described. Now, we draw a certain point in the $feasible\ region$ of Figure.4(a), which is shown as the red point in Figure.4(a). In this case, we call the red point the $working\ point$, which means the cloud platform is operating with the server's size and arrival rate being set as the horizontal and vertical coordinates of this point respectively. When perturbation happens, the server's size or the arrival rate may deviate from the $working\ point$. If the deviation degree is small, the new $working\ point$ will still be located

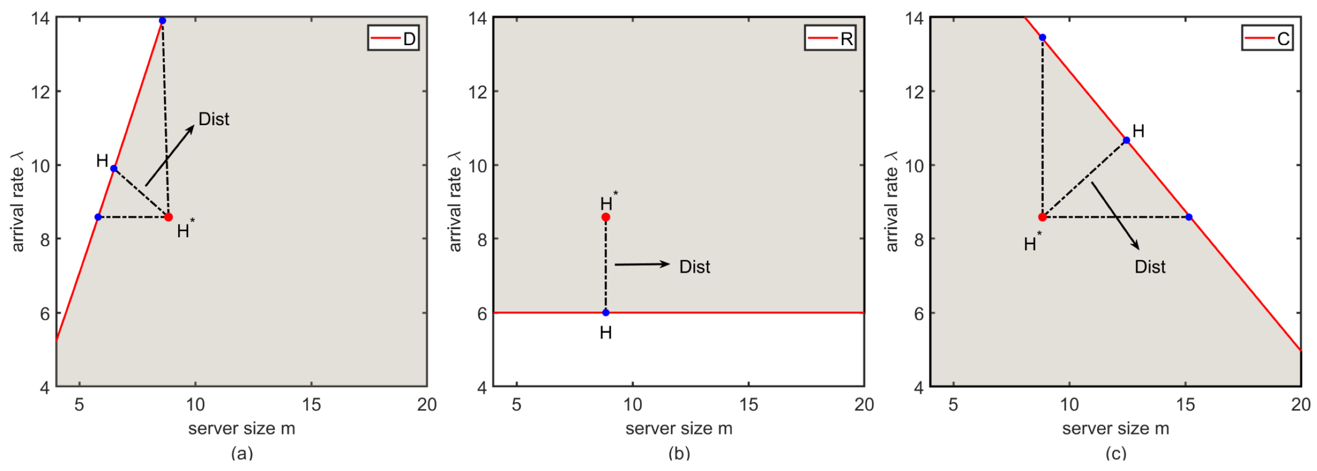

Fig. 4: (a)The $working\ point$ H_i^* versus to D_i. (b)The $working\ point$ H_i^* versus to R_i. (c)The $working\ point$ H_i^* versus to C_i.

in the $feasible\ region$. However, if the deviation degree tends to increase, the waiting time of customers may be greater than the deadline, which is unacceptable for the customers.

Additionally, specific $working\ points$ are marked in Figure.4(b) and Figure.4(c) in the same way. In both cases, when disturbances occur, the server's size and arrival rate will deviate from the $working\ point$. If the deviation is small, the new $working\ point$ will still be within the $feasible\ region$. However, if the deviation increases, it may lead to the cloud service provider's revenue falling below its acceptable minimum or costs exceeding its acceptable maximum, both of which are unacceptable for the cloud service providers.

To overcome the above problems for customers and cloud service providers, by introducing the idea of robustness, we try to promote the performance of the cloud platform as much as possible when it is operating in the worst case. On this basis, any other cases better than the current one will lead to better performance. Specifically, consider Figure.4(a), Figure.4(b) and Figure.4(c) respectively, we can denote the $working\ point$ as H_i^*, and any point on the curves as H_i, then the distance between H_i^* and H_i can be defined as follow.

$$Dist_i = \|H_i^* - H_i\|_2 \tag{23}$$

Further, for any $working\ point$ within the $feasible\ region$ in Figure.4(a), Figure.4(b), and Figure.4(c), we can find the shortest distance between it and any point on the curves. In this case, due to deviations in server size and arrival rate, we consider that when disturbances occur, the probability of the cloud platform operating in an unacceptable region is highest. Therefore, we need to find the optimal $working\ point$ within the $feasible\ region$ to maximize the shortest distance among the distances between that point and the points on the curves.

B. Robustness metric

Now, we devote ourselves to calculating the shortest distances from the $working\ point$ to the boundaries concerning the waiting time, revenue, and cost, respectively. The principle for searching the optimal $working\ point$ is maximizing the previously mentioned shortest distance, which is similar to the idea of robustness described in the control field. Hence, it is reasonable to rename such a distance as the robustness radius [28], which can be represented as follows.

$$r_i^* = min\{Dist\} = min\|H^* - H\|_2 \tag{24}$$

Given a specific $working\ point$ H^* in $feasible\ region$, it is easy to observe that due to the complexity of the formulas for the waiting time, revenue, and cost of cloud service providers, as well as the waiting time of customers, it is challenging to derive a mathematical expression for the shortest distance from that point to the boundary. Therefore, we assume that a circle whose center is H^* and radius is r^* will be tangent to such a boundary. In this case, we propose an adaptive adjustment scheme for the radius r^*, such that the circle centered on H^* can be tangent to the boundary. The pseudo-code for the scheme is shown in Algorithm.1 [29]. The following $\mathbb{1}_1$ and $\mathbb{1}_2$ represent indicator functions.

$$\mathbb{1}_1 = \begin{cases} 0, & if \quad l(x,y) = g(x,y) \\ 1, & \qquad else \end{cases} \tag{25}$$

$$\mathbb{1}_2 = \begin{cases} 0, & if \quad l(x,y) = f(x,y) \\ 1, & \qquad else \end{cases} \tag{26}$$

V. OPTIMAL ROBUST CONFIGURATION

Based on the above discussion, to keep the reduction of disturbance factors on system performance within a controllable range, we propose a robustness configuration scheme in this section. In subsection V-A, we preliminarily consider any one of the subsystems within the parallel mutual exclusion system and propose a reasonable robust configuration scheme for the subsystem. In subsection V-B, comprehensively considering the robustness requirement of the whole parallel mutual system, we propose an appropriate robust configuration scheme that can achieve a balanced improvement of the performance in all subsystems.

A. The subsystem

In the previous section, we propose an algorithm to obtain the robust radius among the distances between a specific

Algorithm 1: Robust radius r^* from *working point* to the boundary

```
Input: Initial position $H^* = (m_z, \lambda_z)$, objective function $l(x,y)$, boundary $c$, $\boldsymbol{\theta}$, $\upsilon_1$, $\upsilon_2$, $\mathbb{1}_1$, $\mathbb{1}_2$, $flag$
Output: Robust radius $r^*$ from working point to the boundary
begin
    $\upsilon_1, \upsilon_2 \leftarrow 10^{-5}$;
    $r_1^* \leftarrow 10^{-4}$;
    Initialization: $x \leftarrow m_z$, $y \leftarrow \lambda_z$;
    $F \leftarrow l(x,y) - c$;
    if $F > 0$ then
        The working point is not within feasible region, $r^* \leftarrow -\infty$;
    else
        $flag \leftarrow 1$;
        while $flag = 1$ do
            for $\theta_i \leftarrow 0$ to $2\pi$ do
                $x_i \leftarrow m_z + r_1^* \cdot \cos\theta_i$;
                $y_i \leftarrow \lambda_z + r_1^* \cdot \sin\theta_i$;
                $F_i \leftarrow l(x_i, y_i) - c$;
            end
            if $\sum_{i=1}^{n} |\mathbf{l}(\mathbf{x_i}, \mathbf{y_i}) - c| \neq \left|\sum_{i=1}^{n} [\mathbf{l}(\mathbf{x_i}, \mathbf{y_i}) - c]\right|$ then
                $r_1^* \leftarrow r_1^* - \mathbb{1}_1 \cdot \upsilon_1 - \mathbb{1}_2 \cdot \upsilon_2$;
                if $r_1^* = 0$ then
                    $flag \leftarrow 0$;
                end
            else if $\prod_{i=1}^{n} \{\boldsymbol{l}(\boldsymbol{x_i}, \boldsymbol{y_i}) - c\} \neq 0$ then
                Find $F_j = 0$, $j \in \{1, 2, \cdots, i\}$;
                $x \leftarrow x_j$, $y \leftarrow y_j$;
                $flag \leftarrow 0$;
            else
                Find the minimum $F_k$, $k \in \{1, 2, \cdots, i\}$;
                $x \leftarrow x_k$, $y \leftarrow y_k$;
                $r_1^* \leftarrow 10^{-4}$;
            end
        end
    end
    $H \leftarrow (x, y)$;
    $r^* \leftarrow \|H^* - H\|_2$;
    return $r^*$
end
```

working point and the points on a given boundary. Through this approach, it is possible to achieve the maximum "shortest distance" as defined in (24). Furthermore, as the parallel system comprises multiple subsystems, for each of them, the goal is to enhance its performance as much as possible when the cloud platform is disturbed. Therefore, in this section, we apply this method to consider the scenario for any subsystem within the parallel system.

Naturally, what can be noted in Figure.5 is that as the *working point* moves closer from the *feasible region*'s boundaries, it may lead to the following scenarios. One is the increase in the waiting time of customers. This raises the likelihood that the waiting time exceeds the deadline. Another is the decrease in the revenue of cloud service providers, which increases the probability of revenue falling below the acceptable minimum revenue under disturbance influence. Or the third case, the cost of the cloud service providers increases. In these cases, it becomes challenging to simultaneously meet the demands of both the cloud service providers and the customers.

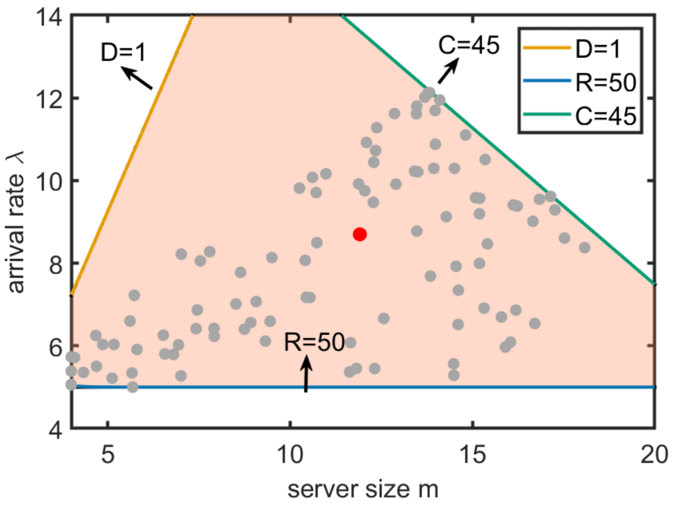

Fig. 5: The acceptable *working points* in the subsystem.

Without loss of generality, the acceptable deadline revenue and cost are set to $D_i = 1$, $R_i = 50$, and $C_i = 45$, respectively. Within the *feasible region* defined by these demands, in order to simultaneously improve the revenue, reduce the cost, and shorten the waiting time, we aim for the optimal *working point* to be as far away as possible from the boundaries formed by these demands.

Therefore, for the problem involving the maximization of the robustness radius, we established a constrained multi-objective optimization model as follows.

$$\max_{m_z, \lambda_z} r^* = (r_1^*, r_2^*, r_3^*)$$
$$s.t. \begin{cases} f(m_z, \lambda_z) < D_i \\ r(m_z, \lambda_z) > R_i \\ c(m_z, \lambda_z) < C_i \\ m_z \in [4, 20], \lambda_z \in [4, 14] \end{cases} \tag{27}$$

In (27), r_1^*, r_2^*, and r_3^* represent the shortest distances from a specific *working point* to the boundaries of waiting time, revenue, and cost, respectively. D_i represents the maximum acceptable waiting time for customers, R_i and C_i represent the minimum acceptable revenue and the maximum acceptable cost for the cloud service providers, respectively. Obviously, the objective of (27) is to maximize r_1^*, r_2^*, and r_3^*, and the constraints indicate that the working point can only be selected within the *feasible region* enclosed by the boundaries.

To solve the multi-objective programming model presented in (27), Non-dominated Sorting Genetic Algorithms II (NSGA-II) is introduced in this paper. The pseudo-code for such an algorithm is shown in Algorithm.2.

By applying the Algorithm.2, the results are shown in Figure.5 and Figure.6. From Figure.5, it can be observed

Algorithm 2: Robust configuration method based on NSGA-II algorithm

```
Input: interval of servers size$[m_{min}, m_{max}]$, servers speed $[\lambda_{min}, \lambda_{max}]$
Output: optimal $r_1^*$, $r_2^*$, $r_3^*$
1  begin
2      $t \leftarrow 1, i \leftarrow 1$;
3      Initialize the population in the given interval;
4      The fitness value $Y$ is calculated through Algorithm 1;
5      $T_1 \leftarrow$ perform non-dominated sorting strategy;
6      Sort $f$ by crowding distance for each rank ;
7      while $t <$ Max number of iterations do
8          $P_t \leftarrow$ create parent by $T_1$ using tournament selected;
9          $Q_t \leftarrow$ create offspring by $P_t$ using selection, crossover and mutation;
10         $R_t \leftarrow P_t \cup Q_t$ ;
11         The fitness value $Y_t$ is calculated through Algorithm 1;
12         $F_t \leftarrow$ calculate all non-dominated fronts of $R_t$;
13         $P_{t+1} \leftarrow \emptyset$ ;
14         while $|P_{t+1}| \cup |F_t^i| \leq N$ do
15             $F_t^i \leftarrow$ select $i$th non-dominated front in $F_t$ using crowding distance sorting strategy;
16             $P_t \leftarrow P_{t+1} \cup F_t^i$;
17             $i \leftarrow i + 1$;
18         end
19         $T_{t+1} \leftarrow P_{t+1} \cup F_t [1 : (N - |P_{t+1}|)]$;
20         $t \leftarrow t + 1$;
21     end
22 end
```

that the gray points represent the considered acceptable set of $working\ points$. In Figure.6, each point on the Pareto front represents an optimal solution of the multi-objective optimization model, corresponding to the robust radii to the boundaries. In general, if it is too saturated to meet one of the requirements of waiting time, revenue, and cost while ignoring the other two requirements, it is not conducive for the subsystem to maintain its performance level under the influence of disturbances. For example, blindly moving the working point as far as possible from the boundary of waiting time, may lead to reduced revenue below the acceptable minimum revenue under the influence of disturbances, or the cost may exceed the acceptable maximum cost. A reasonable approach is to keep the robustness radius as equal as possible for customer's waiting time, cloud service providers' revenue, and cost, allowing them to simultaneously meet the robustness requirements against disturbances. We use the following (28) to assess the deviations between the robustness radius of each obtained acceptable $working\ point$. This allows us to make an appropriate selection and identify the optimal $working\ point$ within the $feasible\ region$.

$$\delta = \frac{\sum_{i=1}^{n} \left(r_i^* - \overline{r}^*\right)^2}{n} \tag{28}$$

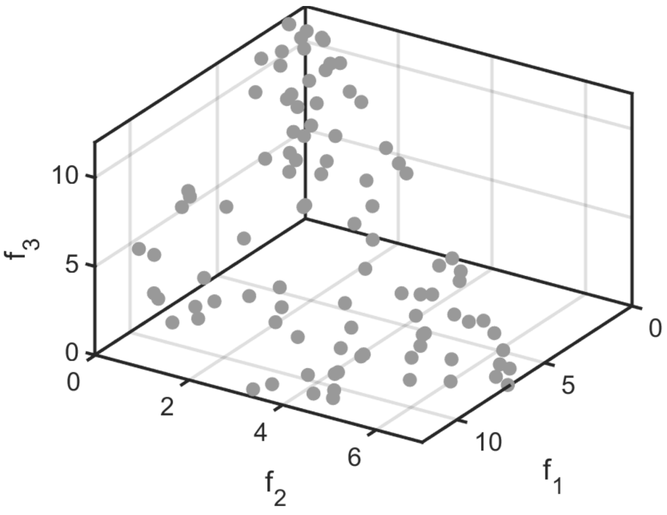

Fig. 6: The Pareto frontier of r_i^* in subsystem.

where $\overline{r}^* = \frac{\sum_{i=1}^{n} r_i^*}{n}$, n represents the number of objectives.

Through (28), we determine the optimal $working\ point$ as the red point in Figure.5. At this point, the selected server size and arrival rate are denoted as $m = 11.91$ and $\lambda = 8.70$, respectively. The shortest robustness radii from this optimal $working\ point$ to the boundaries are $r_1^* = 6.45$, $r_2^* = 3.70$, and $r_3^* = 3.92$.

B. Parallel mutual exclusion system

Through the above discussion on robustness analysis, we understand that for any subsystem, it is a challenge to balance the customer's expectation of the shortest waiting time and the cloud service provider's goal of maximizing revenue and minimizing costs. On this basis, we further analyze the optimal robust configuration problem for the entire parallel system, taking into account the impact of specific perturbations on cloud service providers and customers in multiple subsystems.

Due to the limited shared resources in a parallel system, each subsystem must compete for the necessary resources. This competition is a common phenomenon because each subsystem needs to meet its demands as much as possible. The competition for resources among subsystems leads to mutual influence and restriction among them, so it is necessary to adopt appropriate strategies to balance resource utilization and performance requirements. Moreover, it is realized that there may be some differences among the lower thresholds of the requirements of the subsystems for their demands. Therefore, we subdivide different scenarios for discussion.

1) Subsystems with the same demands: In this subsection, we analyze scenarios where the subsystems within a parallel mutual exclusion system share identical lower thresholds for deadlines, revenue, and costs. The goal for each subsystem is to fulfill customer demands within a specified deadline while minimizing costs and maximizing revenue. However, fluctuations in server size and arrival rates can disrupt this balance. For instance, an unexpected change in server capacity or an increase in arrival rates can extend customer waiting times, potentially causing them to exceed the deadline.

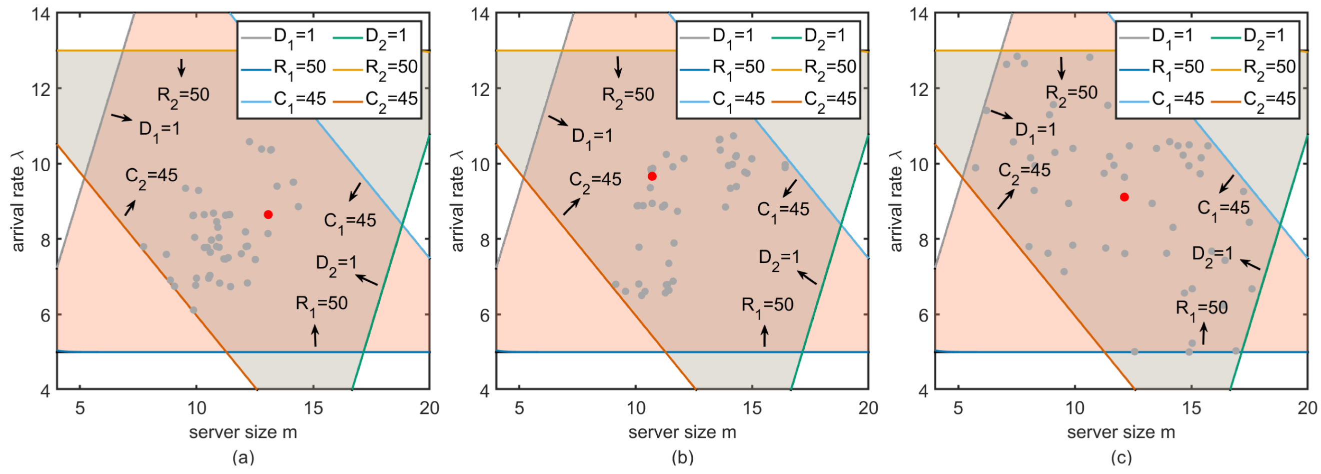

Fig. 7: (a) The acceptable *working points* in system obtained by OptMPNDS. (b) The acceptable *working points* in system obtained by OptMPNDS2. (c) The acceptable *working points* in system obtained by OptMPNDS3.

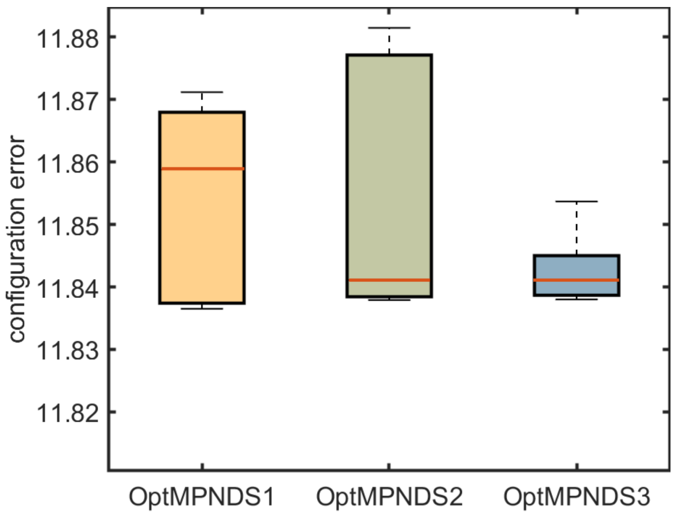

Fig. 8: The acceptable *working points* with different robustness radius error.

Figure.7 illustrates the *feasible region* delineated by the acceptable lower thresholds for demand in each subsystem. The orange area represents the *feasible region* for subsystem S_1, while the gray area represents that of subsystem S_2. The overlapping portion of these two regions represents the true *feasible region* that should be considered when configuring the overall cloud service system.

To aid in the discussion, we assume the total server capacity available for allocation in the parallel mutual exclusion system is $m_{all} = 24$, and the total customer arrival rate is $\lambda_{all} = 18$. Given that the subsystems share consistent lower demand thresholds, the acceptable upper limits for deadlines, minimum revenue, and maximum cost in subsystems S_1 and S_2 are set as $D_i = 1$, $R_i = 50$, and $C_i = 45$, respectively, for both $i = 1, 2$.

Under perturbation conditions, subsystems S_1 and S_2 prioritize the demands of their respective cloud service providers and customers. This resource allocation challenge, involving multiple decision-makers with distinct objectives, can be framed as a multi-party multi-objective optimization problem. Specifically, the goal is to identify the optimal *working point* within the defined *feasible region* that maximizes the satisfaction of each subsystem's requirements. To achieve this, we formulate a multi-party multi-objective optimization model with constraints reflecting the actual demands of each subsystem, as described below.

$$\begin{aligned}
&\max F = (F_1, ..., F_i), i = 2 \text{ } or \text{ } 3\\
&F_j = (r_1^*, ..., r_j^*)\\
&s.t. \begin{cases} f(m_{zj}, \lambda_{zj}) < D_i \\ r(m_{zj}, \lambda_{zj}) > R_i \\ c(m_{zj}, \lambda_{zj}) < C_i \\ m_{zj} \in [4, 20], \lambda_{zj} \in [4, 14] \end{cases}
\end{aligned} \tag{29}$$

To solve the multi-party multi-objective optimization model proposed in (29), we introduce a robust configuration method based on the OptMPNDS3 algorithm (i.e., Algorithm 3) [30]. This algorithm is designed to search for the optimal *working point* within the *feasible region*, taking into account the requirements of each subsystem. The results of this search are illustrated in Figure.7(c), where the gray points represent the set of working points acceptable to both subsystems S_1 and S_2.

When dealing with uncertain perturbations, it is generally inadvisable for the cloud platform to prioritize the requirements of one subsystem at the expense of others. In other words, while setting the optimal *working point* for a subsystem as far from the boundary as possible, the robustness requirements of the other subsystems should not be overlooked. To address this, we employ (28) to systematically evaluate the robustness trade-offs among the gray *working points*, with the objective of identifying the optimal *working point* for the entire parallel mutual exclusion system. For instance, by applying the aforementioned method, we can determine the optimal *working point* for the cloud platform, which is represented by the red *working point* in Figure 7(c).

Figure.7(a) and (b) show the optimal *working points* obtained by OptMPNDS [31] and OptMPNDS2 [32], respectively, that is, the red points in the figure. Through 20 sets of tests on the three methods independently, the formula (28) is used to evaluate each group and select the *working points* obtained by the experiment. The experimental results are shown in Figure.8. In the absence of outliers, the box plot

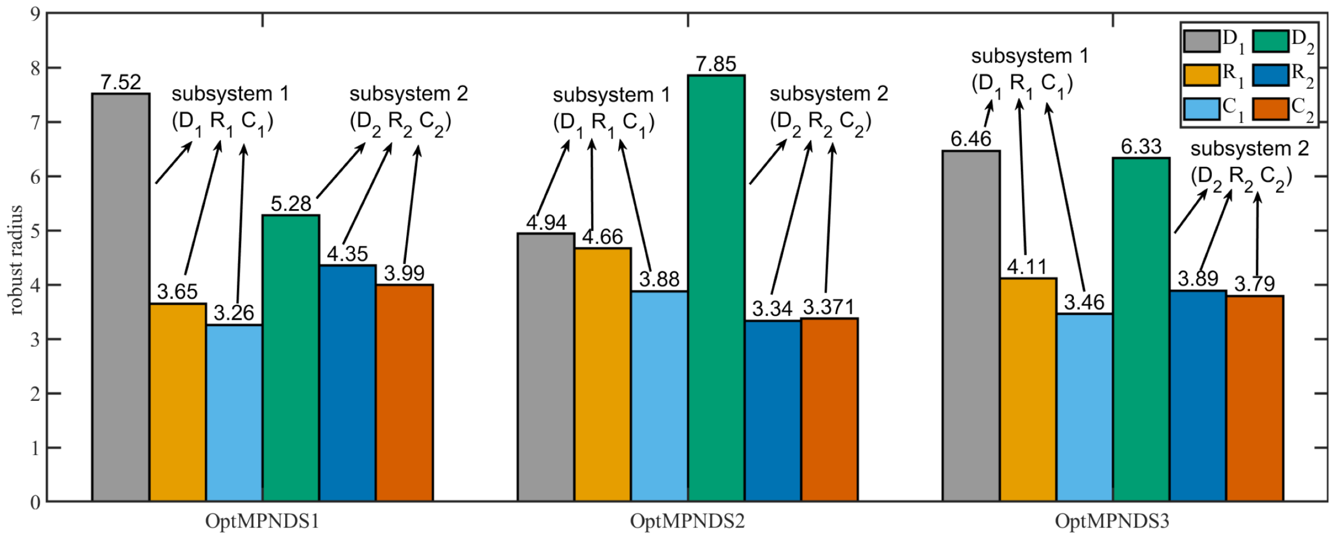

Fig. 9: Robustness radius r_i^* of each subsystem obtained by different methods

shows that the stability of the robust configuration method based on the OptMPNDS3 algorithm is much better than other comparison methods. For example, using the above method, we can obtain the optimal $working\ point$ of the cloud platform, that is the red $working\ point$ in Figure.7(c).

Therefore, we also select the $working\ point$ with the smallest robustness radius error in our proposed configuration method, that is, the server size and arrival rate in the subsystem s_1 are $m_1 = 12.13$ and $\lambda_1 = 9.11$, respectively. Correspondingly, the server size in the subsystem S_2 is $m_2 = 11.87$, and the arrival rate is $\lambda_2 = 8.89$. As shown in Figure 9, each bar graph represents the robustness radius of subsystems S_1 and S_2 in terms of deadline, revenue, and cost obtained by different algorithms. By configuring the optimal $working\ point$ of the cloud platform, the robustness radius of the deadline, revenue and cost of the subsystem S_1 are $r_1^* = 6.46$, $r^*2 = 4.11$ and $r_3^* = 3.46$, respectively. Correspondingly, the robustness radius of the subsystem S_2 is $r_1^* = 6.33$, $r^*2 = 3.89$, $r_3^* = 3.79$. The figure clearly shows that the OptMPNDS3 algorithm achieves a smaller absolute error in the robustness radius under all requirements, indicating that it has higher efficiency in balancing the demands of each subsystem. That is to say, the configuration method significantly enhances the robustness of the subsystem in the face of perturbations and achieves a better balance between deadline, revenue, and cost.

Through the analysis of Figure.7 and Figure.9, it can be seen that the robustness configuration scheme has significant advantages in optimizing the robust configuration of subsystems with the same demands in parallel mutual exclusion systems. This scheme better balances the demands of each subsystem, enhancing the overall performance and robustness of the system under disturbances.

2) Subsystems with the different demands: In parallel mutual exclusion systems, varying preferences between customers and cloud service providers can result in differentiated performance requirements across subsystems. For instance, some subsystems may prioritize delivering high-quality services, while others may focus on maximizing throughput. This leads to distinct performance expectations, with certain subsystems imposing more stringent demands

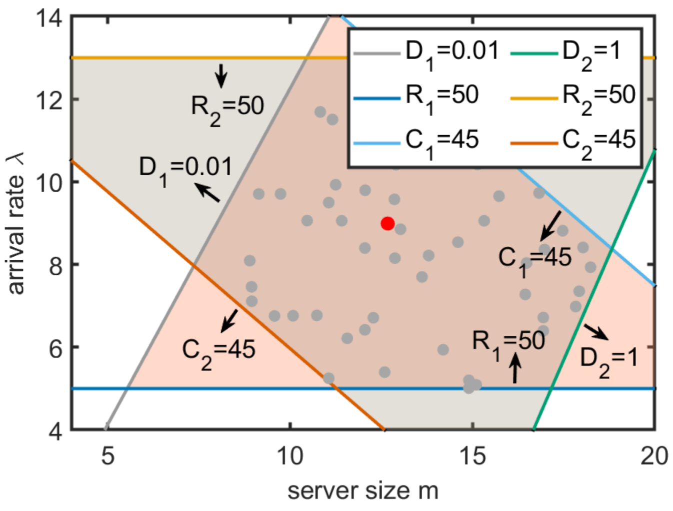

Fig. 10: The acceptable $working\ points$ with few different demands between subsystems.

on specific performance metrics. Given the prevalence of these differentiated requirements, it is essential to address the challenge of robust configuration in such cases.

Building on the problem outlined above, we first examine the scenario depicted in Figure.10. In this case, when the parallel system processes tasks, subsystem S_1 imposes stricter requirements on the service response deadline compared to subsystem S_2, with $D_1 = 0.01$, $R_1 = 50$, and $C_1 = 45$, while for subsystem S_2, the corresponding values are $D_2 = 1$, $R_2 = 50$, and $C_2 = 45$. To address this, we apply the OptMPNDS3 algorithm, as described earlier, to identify acceptable $working\ points$ within the $feasible\ region$ and select the optimal configuration scheme using (28).

When the server size and arrival rates for S_1 and S_2 are configured based on the optimal $working\ point$, the numerical results are summarized in Table II. For subsystem S_1, the server size is $m_1 = 12.68$ and the customer arrival rate is $\lambda_1 = 8.99$. Under this configuration, the robustness radii for the deadline, revenue, and cost are $r_1^* = 3.98$, $r_2^* = 3.99$, and $r_3^* = 3.22$, respectively. For subsystem S_2, the corresponding values are $m_2 = 11.32$, $\lambda_2 = 9.01$, with

Algorithm 3: Robust configuration method based on OptMPNDS3 algorithm

```
   Input: $N$(the population size),
          $F \leftarrow (F_1(x), F_2(x), ..., F_M(x))$(the objective
          function)
   Output: $MPS$ (the multiparty Pareto optimal
           solutions)
1  begin
2      Initialization population $P_0$;
3      $F$ is calculated through Algorithm 1;
4      $t \leftarrow 0, Q_t \leftarrow \emptyset, A \leftarrow \emptyset$;
5      $\mathcal{L} \leftarrow$ NondominatedSorting$(P_0, F)$;
6      $P_0 \leftarrow$ NondominatedSorting$(P_0, \mathcal{L})$;
7      Sort $P_0$ by crowding entropy for each rank ;
8      while $FE$ is not reached do
9          Offspring $Q_t$ are generated from $P_t$ and $A$
            using parameters $u_{CR}$ and $u_F$;
10         The non-dominated solutions in $Q_t$ and $P_t$ are
            storied in $R_t$, others are put into $A$;
11         $\mathcal{L} \leftarrow$ NondominatedSorting$(P_0, F)$;
12         $B \leftarrow$ NondominatedSorting$(P_0, \mathcal{L})$;
13         Sort $B$ by crowding entropy for each rank ;
14         $i \leftarrow 1, P_{t+1} \leftarrow \emptyset$;
15         while $|P_{t+1} \cup B_i| \leq N$ do
16             $P_{t+1} \leftarrow P_{t+1} \cup B_i$;
17             $i \leftarrow i + 1$;
18         end
19         while $|P_{t+1}| < N$ do
20             Move the least crowded solution from $B_i$
                into $P_{t+1}$;
21             Update the crowding entropy of $B_i$;
22         end
23         Update $A$ with individuals not selected;
24         Put $CR$ and $F$ which generate solutions in
            $P_{t+1}$ into $S_{CR}$ and $S_F$;
25         Update the parameters $u_{CR}$ and $u_F$ with $S_{CR}$
            and $S_F$;
26         $t \leftarrow t + 1$;
27     end
28     Set $MPS$ as solutions which is Pareto optimal for
        all parties in $P_t$;
29 end
```

TABLE II: The optimal robust solution obtained with few different demands between subsystems.

S_i	Lowest demand			Working point		Robust radius		
	D_i	R_i	C_i	m_i	λ_i	r_1^*	r_2^*	r_3^*
S_1	0.01	50.00	45.00	12.68	8.99	3.98	3.99	3.22
S_2	1.00	50.00	45.00	11.32	9.01	5.78	4.01	4.02

robustness radii of $r_1^* = 5.78$, $r_2^* = 4.01$, and $r_3^* = 4.02$.

As illustrated in Figure 10, the red point represents the optimal $working\ point$ selected within the $feasible\ region$. The performance metrics for both subsystems meet their respective minimum requirements, and the deadlines for S_1 and S_2 are optimized. These results demonstrate that our method effectively addresses the challenge of optimizing subsystems with differentiated performance demands.

Then, we consider more complex cases, such as Figure.11.

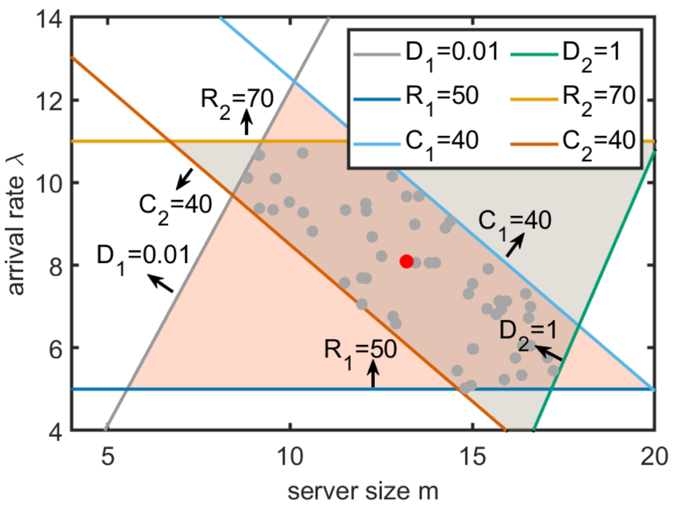

Fig. 11: The acceptable $working\ points$ with numerous different demands between subsystems.

In the parallel system, subsystem S_1 aims to respond to customer service requests after a short deadline, while subsystem S_2 is committed to bringing high revenue to cloud service providers, resulting in demand differences (i.e., D_1 = 0.01, R_1 = 50, C_1 = 40, D_2 = 1, R_2 = 70, C_2 = 40). To solve this problem, we also use our proposed method to obtain the acceptable $working\ points$ in the $feasible\ region$ and obtain the optimal configuration scheme through (28).

TABLE III: The optimal robust solution obtained with numerous different demands between subsystems.

S_i	Lowest demand			Working point		Robust radius		
	D_i	R_i	C_i	m_i	λ_i	r_1^*	r_2^*	r_3^*
S_1	0.01	50.00	40.00	13.19	8.09	4.88	3.09	1.61
S_2	1.00	70.00	40.00	10.81	9.91	4.92	2.91	1.61

Through the configuration results shown in Figure.11, it can be found that the two subsystems have the same robustness in the face of perturbations. This is because when determining the optimal $working\ point$, a compromise solution is adopted, that is, not favoring either side. Under the optimal configuration, the calculation of the robustness radius of S_1 and S_2 is recorded in Table III. It can be seen that the server size and customer arrival rate of S_1 are m_1 = 13.19, λ_1 = 8.09, and the robustness radius of deadline, revenue and cost are r_1^* = 4.88, r_2^* = 3.09, r_3^* = 1.61, respectively. For S_2, the corresponding values are m_2 = 10.81, λ_2 = 9.91, and r_1^* = 4.92, r_2^* = 2.91, r_3^* = 1.61. Compared with the two subsystems, its performance meets the minimum demand, the deadline of S_1 is greatly shortened, and the revenue of S_2 is greatly increased. This shows that our configuration scheme can not only ensure the robustness of the entire parallel system but also meet the differentiated demands of each subsystem.

VI. Conclusion

This paper presents a novel robust configuration method for parallel mutual exclusion systems within cloud computing environments to effectively address the impact of uncertain disturbances on system performance. By viewing the system as a collection of multi-server queuing models and considering key performance metrics such as customer waiting time, cloud service providers' revenue, and cost, an in-depth robustness analysis for each subsystem is conducted.

Furthermore, these robustness analysis methods are extended to the entire parallel system, addressing the competition and mutual constraints among subsystems to ensure overall system robustness when facing disturbances. A key contribution of this paper is the introduction of the robustness radius metric, which numerically characterizes system robustness. This measure aids in designing a robustness optimization problem based on a multi-party multi-objective model, solved using the OptMPNDS3 algorithm, assisting decision-makers in making informed choices.

Experimental results validate the proposed robust configuration scheme, demonstrating that it not only ensures system performance but also maximally satisfies the demands of both cloud service providers and customers. The parameter settings and optimal solutions obtained in this study have practical implications for enhancing the robustness and performance of parallel mutual exclusion systems. The findings provide valuable references for further research on the robustness of such systems, contributing to the advancement of cloud computing resource management practices.

References

[1] J. Jin, R. Li, X. Yang, M. Jin, and F. Hu, "A network slicing algorithm for cloud-edge collaboration hybrid computing in 5g and beyond networks," *Computers and Electrical Engineering*, vol. 109, p. 108750, 2023.

[2] G. Zheng, H. Zhang, Y. Li, and L. Xi, "5g network-oriented hierarchical distributed cloud computing system resource optimization scheduling and allocation," *Computer Communications*, vol. 164, pp. 88–99, 2020.

[3] M. A. Al-Sharafi, M. Iranmanesh, M. Al-Emran, A. I. Alzahrani, F. Herzallah, and N. Jamil, "Determinants of cloud computing integration and its impact on sustainable performance in smes: An empirical investigation using the sem-ann approach," *Heliyon*, vol. 9, no. 5, 2023.

[4] Y. S. Jghef, S. Zeebaree *et al.*, "State of art survey for significant relations between cloud computing and distributed computing," *International Journal of Science and Business*, vol. 4, no. 12, pp. 53–61, 2020.

[5] V. Meyer, M. L. da Silva, D. F. Kirchoff, and C. A. De Rose, "Iada: A dynamic interference-aware cloud scheduling architecture for latency-sensitive workloads," *Journal of Systems and Software*, vol. 194, p. 111491, 2022.

[6] V. Meyer, D. F. Kirchoff, M. L. Da Silva, and C. A. De Rose, "Ml-driven classification scheme for dynamic interference-aware resource scheduling in cloud infrastructures," *Journal of Systems Architecture*, vol. 116, p. 102064, 2021.

[7] C. K. Swain and A. Sahu, "Interference aware workload scheduling for latency sensitive tasks in cloud environment," *Computing*, vol. 104, no. 4, pp. 925–950, 2022.

[8] L. Zhao, Y. Hu, X. Yang, Z. Dou, and L. Kang, "Robust multi-task learning network for complex lidar point cloud data preprocessing," *Expert Systems with Applications*, vol. 237, p. 121552, 2024.

[9] J. Fang, G. Zhao, H. Xu, H. Tu, and H. Wang, "Reveal: Robustness-aware vnf placement and request scheduling in edge clouds," *Computer Networks*, vol. 233, p. 109882, 2023.

[10] J. Wang and Y. Wang, "An efficient improved whale optimization algorithm for optimization tasks." *Engineering Letters*, vol. 32, no. 2, pp. 392–411, 2024.

[11] X. Wang, H. Lou, Z. Dong, C. Yu, and R. Lu, "Decomposition-based multi-objective evolutionary algorithm for virtual machine and task joint scheduling of cloud computing in data space," *Swarm and Evolutionary Computation*, vol. 77, p. 101230, 2023.

[12] M. Qin, M. Li, and R. O. Yahya, "Dynamic iot service placement based on shared parallel architecture in fog-cloud computing," *Internet of Things*, vol. 23, p. 100856, 2023.

[13] A. V. Gorbunova and V. M. Vishnevsky, "Estimating the response time of a cloud computing system with the help of neural networks," *Advances in Systems Science and Applications*, vol. 20, no. 3, pp. 105–112, 2020.

[14] R. Wang, G. Zai, Y. Liu, and H. Pang, "Service performance analysis of cloud computing server by queuing system," in *International Conference on Mobile Computing, Applications, and Services*. Springer, 2021, pp. 42–53.

[15] S. Chen, J. Liu, F. Ma, and H. Huang, "Customer-satisfaction-aware and deadline-constrained profit maximization problem in cloud computing," *Journal of Parallel and Distributed Computing*, vol. 163, pp. 198–213, 2022.

[16] S. Lv, M. Yang, and J. Wen, "The m/m/1 repairable queueing system with two types of server breakdowns and negative customers." *Engineering Letters*, vol. 32, no. 1, pp. 77–83, 2024.

[17] S. Souravlas, S. Katsavounis, and S. Anastasiadou, "On modeling and simulation of resource allocation policies in cloud computing using colored petri nets," *Applied Sciences*, vol. 10, no. 16, p. 5644, 2020.

[18] D. Jia, M. Bayati, R. Lee, and N. Mi, "Rita: efficient memory allocation scheme for containerized parallel systems to improve data processing latency," in *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*. IEEE, 2020, pp. 329–336.

[19] K. Karthiban and J. S. Raj, "An efficient green computing fair resource allocation in cloud computing using modified deep reinforcement learning algorithm," *Soft Computing*, vol. 24, no. 19, pp. 14 933–14 942, 2020.

[20] H. Yuan and M. Zhou, "Profit-maximized collaborative computation offloading and resource allocation in distributed cloud and edge computing systems," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1277–1287, 2020.

[21] S. Rostami, A. Broumandnia, and A. Khademzadeh, "An energy-efficient task scheduling method for heterogeneous cloud computing systems using capuchin search and inverted ant colony optimization algorithm," *The Journal of Supercomputing*, vol. 80, no. 6, pp. 7812–7848, 2024.

[22] R. Stewart, A. Raith, and O. Sinnen, "Optimising makespan and energy consumption in task scheduling for parallel systems," *Computers & Operations Re-*

search, vol. 154, p. 106212, 2023.

[23] R. M. Abdelmoneem, A. Benslimane, and E. Shaaban, “Mobility-aware task scheduling in cloud-fog iot-based healthcare architectures,” *Computer networks*, vol. 179, p. 107348, 2020.

[24] J. Mei, K. Li, Z. Tong, Q. Li, and K. Li, “Profit maximization for cloud brokers in cloud computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 1, pp. 190–203, 2018.

[25] S. Chen, S. Huang, Q. Luo, and J. Zhou, “A profit maximization scheme in cloud computing with deadline constraints,” *IEEE Access*, vol. 8, pp. 118 924–118 939, 2020.

[26] Z. Li, V. Chang, H. Hu, D. Yu, J. Ge, and B. Huang, “Profit maximization for security-aware task offloading in edge-cloud environment,” *Journal of Parallel and Distributed Computing*, vol. 157, pp. 43–55, 2021.

[27] T. Wang, J. Zhou, L. Li, G. Zhang, K. Li, and X. S. Hu, “Deadline and reliability aware multiserver configuration optimization for maximizing profit,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 3772–3786, 2022.

[28] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim, “Measuring the robustness of a resource allocation,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 7, pp. 630–641, 2004.

[29] J. Zhou, S. Chen, and H. Kuang, “Robust scheduling in cloud environment based on heuristic optimization algorithm,” *arXiv preprint arXiv:2311.17757*, 2023.

[30] Z. She, W. Luo, X. Lin, Y. Chang, and Y. Shi, “Multiparty distance minimization: Problems and an evolutionary approach,” *Swarm and Evolutionary Computation*, vol. 83, p. 101415, 2023.

[31] W. Liu, W. Luo, X. Lin, M. Li, and S. Yang, “Evolutionary approach to multiparty multiobjective optimization problems with common pareto optimal solutions,” in *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2020, pp. 1–9.

[32] Z. She, W. Luo, Y. Chang, X. Lin, and Y. Tan, “A new evolutionary approach to multiparty multiobjective optimization,” in *International Conference on Swarm Intelligence*. Springer, 2021, pp. 58–69.