Multi-Queue Request Scheduling for Profit Maximization in IaaS Clouds

Shuang Wang, Xiaoping Li, *Senior Member, IEEE*, Quan Z. Sheng, Rubén Ruiz, Jinquan Zhang, and Amin Beheshti

Abstract—In cloud computing, service providers rent heterogeneous servers from cloud providers, i.e., Infrastructure as a Service (IaaS), to meet requests of consumers. The heterogeneity of servers and impatience of consumers pose great challenges to service providers for profit maximization. In this article, we transform this problem into a multi-queue model where the optimal expected response time of each queue is theoretically analyzed. A multi-queue request scheduling algorithm framework is proposed to maximize the total profit of service providers, which consists of three components: *request stream splitting*, *requests allocation*, and *server assignment*. A request stream splitting algorithm is designed to split the arriving requests to minimize the response time in the multi-queue system. An allocation algorithm, which adopts a one-step improvement strategy, is developed to further optimize the response time of the requests. Furthermore, an algorithm is developed to determine the appropriate number of required servers of each queue. After statistically calibrating parameters and algorithm components over a comprehensive set of random instances, the proposed algorithms are compared with the state-of-the-art over both simulated and real-world instances. The results indicate that the proposed multi-queue request scheduling algorithm outperforms the other algorithms with acceptable computational time.

Index Terms—Profit maximization, consumer impatience, queue, scheduling, cloud computing

1 Introduction

Service providers offer resources to service consumers by renting resources from cloud providers such as Amazon EC2, Alibaba Cloud, and Microsoft Azure. Service consumers often put up with a short wait [1], i.e., a service consumer becomes *impatient* if the waiting time exceeds his/her MWT (maximum waiting time) [2], [3]. The profit of an agent increases if requests of consumers are met before their MWTs and decreases when consumers leave. Different service consumers have different impatience levels, implying that MWTs vary, e.g., MWT is *stochastic* with the exponential distribution [4]–[6].

Impatient consumers are common in practice. For example, consumers send requests to a call center [7], [8] which rents servers from Alicloud to fulfill the requests. If service consumers cannot obtain responses within their MWTs, they will leave the call center and their satisfaction would decrease. The most common resource instance provisioning alternatives are: *reserved*, *on-demand*, and *spot*. The rental costs of reserved instances are much lower than those of on-demand but with much longer rental duration. Though spot instances have much lower costs than the reserved ones by bidding instances, i.e., the user paying the highest price obtains the instance, the risk of being interrupted or preempted by a higher bid might lead to late completion or even higher costs due to multiple rentals. Different rental instances result in different prices. Therefore, how to select suitable servers is crucial for dealing with impatient service consumers to maximize the total profit.

In this paper, we consider the problem of scheduling independent requests with distinct MWTs to heterogeneous servers to maximize the total profit of the service agent. Requests arrive at the system stochastically. The heterogeneity of the servers means that servers have different rates which further result in different server configurations and prices. To maximize the total profit, the service agent rents resources from cloud providers by reserved and on-demand instances. In this paper, we do not consider the spot renting alternative because the MWTs of customers would be violated in the case of failed renting with limited bidding prices. If requests are processed within the MWT, these requests are executed by reserved instances and are computed by on-demand instances otherwise.

The considered problem is complicated because of the stochastically arriving requests from often impatient consumers, heterogeneous servers and different types of rented instances. Requests of impatient consumers in distributed systems are processed by a multi-queue system [9], [10] that is common in real scenarios. Each queue contains homogeneous servers, i.e., the same kind of servers are included in the same queue. The number of heterogeneous servers and that of their types

- *Shuang Wang and Xiaoping Li are with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China, and also with the Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, Nanjing 211189, China. E-mail: {wangshuang, xpli}@seu.edu.cn.*
- *Quan Z. Sheng and Amin Beheshti are with the Department of Computing, Macquarie University, Sydney, NSW 2109, Australia. E-mail: {michael.sheng, amin.beheshti}@mq.edu.au.*
- *Rubén Ruiz is with the Grupo de Sistemas de Optimización Aplicada, Universitat Politècnica de València, Camino de Vera s/n, 46022 València, Spain. E-mail: rruiz@eio.upv.es.*
- *Jinquan Zhang is with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China. E-mail: 15895906975@163.com.*

Manuscript received 6 Oct. 2020; revised 8 Mar. 2021; accepted 19 Apr. 2021. Date of publication 23 Apr. 2021; date of current version 25 May 2021.
(Corresponding author: Xiaoping Li.)
Recommended for acceptance by R. Tolosana.
Digital Object Identifier no. 10.1109/TPDS.2021.3075254

are closely related to the total cost of the rented instances and the response time of requests. The bigger the number of heterogeneous servers, the higher the total rental cost. Moreover, less reserved servers lead to more requests unsatisfied and more on-demand servers needed to improve the response time with higher rental costs. The main challenges of the problem under this study are:

(i) How to segment the incoming requests into queues corresponding to heterogeneous servers in order to minimize the expected response time,
(ii) How to allocate tasks to specific queues in order to minimize the number of impatient consumers or to minimize the actual response time, and finally
(iii) How many reserved and on-demand servers have to be rented for the stochastically arriving requests in a period to maximize the total profit.

Over-reserved instances translate into more idle servers while under-reserved instances lead to more impatient consumers. Furthermore, over-rental of on-demand servers incur more rental cost whereas under-rental on-demand servers lead to leaving of more consumers.

We first propose a mathematical model for the considered problem. The MQRS (Multi-Queue Request Scheduling) algorithm framework is then proposed which consists of three phases: (i) splitting the request arrival rate λ into S queues, (ii) allocating requests of each queue to servers, and (iii) deriving the number of rental servers. The main contributions of our work are summarized as follows:

(i) By constructing the $M/M/n_i^r/n_i^r + R$ model for each queue Q_i, the optimal expected response time can be theoretically analyzed,
(ii) Based on the proved properties of the stochastic system, a request stream splitting algorithm is proposed to appropriately split the request arrival rate λ into pieces, and
(iii) To minimize the actual response time of requests, an allocation strategy is introduced to allocate arriving requests to the appropriate queues.

The rest of the paper is organized as follows. The related work is reviewed in Section 2. Section 3 details the model and problem formulation. The optimization methods are proposed in Section 4. The experimental results are reported in Section 5, followed by conclusions and future research direction discussions in Section 6.

2 Related Work

Handing requests from impatient consumers has attracted a considerable research attention recently [2], [11]–[14]. Cost minimization was studied under deadline constraints [11], [12], [14]. Requests have to be processed before the deadlines. A multiple priority preemptive $M/G/1/./EDF$ model for requests was proposed in [12] which processes requests with the earliest deadlines. An $M/M/m + D$ queuing model was constructed with requests and a constant MWT (maximum waiting time) D for profit maximization [2]. The deadline factor in [12] is used to describe the impatience levels of different consumers. All the consumers have the same MWT in [2]. In this queuing model, the deadlines and MWT are constant [2], [11]–[14]. However, it is common in reality for different service consumers to have different levels of impatience. Unfortunately, very few research works consider requests with stochastic MWT for impatient consumers in cloud computing.

Developing an appropriate queue model dealing with requests arriving stochastically has been considered [15]–[18], [23]. Specifically, a multi-queue system processing the requests has been studied in [9], [10], [19]–[23]. The constructed model in [9], [10], [19]–[21] was considered differently from [15]–[18]. A dynamic assignment of impatient customers to multi-queue systems using Markov process technology was analyzed in [9]. Two different queue types were studied in [9]. The system throughput was analyzed using the Markov decision process for assigning services to customers in [10]. Two parallel queues were analyzed for requests in [19]. Near-optimal policies for multi-queue based on wait and performance objectives were studied in [20]. A distributed algorithm was proposed to balance the performance and power consumption for heterogeneous servers [21] for a multi-queue system. The performance was optimized with power constraint in [22]. A G/G/1 queuing system was adopted to analyze the performance of servers in distributed green cloud systems [23]. A geography-aware task scheduling approach was proposed by considering spatial variations in distributed green data centers to maximize the total profit by intelligently scheduling tasks of all applications [24]. According to the constructed multiple queue models, different objectives were considered [10], [20]–[24]. In this paper, we consider the multi-queue for requests in terms of impatient customers for maximizing the total profit, which was not considered in [9], [10], [19]–[21].

Profit maximization for service providers in cloud computing was studied in [2], [25]–[30]. An $M/M/m + D$ queueing model was constructed in [2] to obtain the optimized configurations for maximizing the total profit. Price bidding strategies by a utility function were proposed to maximize the total profit in [25]. Customer satisfaction was found to be a major factor to affect the total profit [26], [27]. A model of cloud networks with different types of servers was constructed by using the auction theory [30]. A profit maximization algorithm was studied aiming to discover the temporal variation of prices in hybrid clouds [28]. A two-stage strategy was proposed to maximize task scheduling performance and minimize non-reasonable task allocation in clouds [29]. Requests processed by multi-queue systems were not studied in [2], [25]–[29]. The stochastic impatience for different consumers was not considered in [27].

Compared to our previous research in [31] that considers heterogeneous servers in a queue, multiple queues for heterogeneous servers are studied in this paper. To the best of our knowledge, the profit maximization problem of renting suitable heterogeneous servers for impatient consumers has never been considered. This makes the studied problem closer to real world applications.

3 System Model and Problem Description

Requests arrive stochastically and independently following a Poisson process with rate λ [6], [32] (e.g., λ of the real requests in Alicloud[1] takes 35.6, 41.8, etc. [31]). Cloud

1. http://clusterdata2018pubcn.oss-cn-beijing.aliyuncs.com/batch_task.tar.gz

TABLE 1
Notations Used in This Paper

Notation	Explanation
λ	Arrival rate of requests
S	Types of servers
Q_i	The i^{th} queue
μ_i	Service rate for the i type server
c_i^r	Unit price for reserved servers with μ_i
c_i^o	Unit price for on-demand servers with μ_i
n_i^r	Number of reserved servers with μ_i
n_i^o	Number of on-demand servers with μ_i
R	Queue capacity in each queue
$\frac{1}{\theta}$	Expected MWT of service consumers
p_j^i	Steady state probability with j requests in Q_i
$\overline{L}(\lambda_i)$	Expected number of requests $L(\lambda_i)$ of Q_i
$\overline{T}_i$	Expected response time of requests with split λ_i
T_i^{os}	Actual response time by allocating requests to Q_i
W_i^j	Waiting time of a new request with j requests in Q_i
P_M^i	Probability of a new request exceeding the MWT
P_i^o	Probability of requests processed on on-demand servers in Q_i
α	Income per request
ω	Time period for the reserved servers

Note: in the table, $i \in \{1, \ldots, S\}$; $j \in \{0, \ldots, n_i^r + R\}$.

providers offer S types of servers and each type is regarded as a queue, i.e., there are S queues $Q_1, \ldots, Q_S$ in the system. There are two phases for scheduling the arriving requests: (i) they are distributed into S queues, and (ii) the requests in each queue are allocated to specific servers. The main notations used in this paper are collected in Table 1.

The processing time of a request by a server is an independent and exponentially distributed random variable [31], [33] where the processing rates satisfy $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_S$. The length of each queue in the system is R. Service providers rent servers from cloud providers with either the reserved or the on-demand alternative. Similar to [41], on-demand servers are rented to improve the system performance if reserved servers cannot meet the requirements of consumers. Corresponding to the server rates μ_1, μ_2, ..., μ_S, the unit prices of reserved servers are $c_1^r, c_2^r, \ldots, c_S^r$ and those of on-demand servers are $c_1^o, c_2^o, \ldots, c_S^o$. The numbers of the required reserved and on-demand servers are $(n_1^r, n_2^r, \ldots, n_S^r)$ and $(n_1^o, n_2^o, \ldots, n_S^o)$, respectively. The income is α after a request is processed. The renting period is ω for each reserved server. Fig. 1 illustrates the system queue model.

The MWT of each service consumer is assumed to be an independent and exponentially distributed random variable with the expected time parameter $\frac{1}{\theta}$.

$$MWT(t) = 1 - e^{-\theta t}. \tag{1}$$

The request stream is split into S small streams (queues) with rates $\lambda_1, \lambda_2, \ldots, \lambda_S$, which follow the Poisson distribution by the Controller as shown in Fig. 1. The Controller dispatches each new request to one of the queues following the first-come, first-served (FCFS) rule. The capacity of each queue is R. Requests are executed with reserved servers except in the following two cases: (i) the queue is full, or (ii) the waiting time of the requests exceeds the MWT. For these two cases, the requests are processed by on-demand servers.

Since both the number of queues and impatience have a great influence on the performance of the system, the theoretical properties of the problem under this study are completely different from those in [31]. Though impatience was also considered in [35], only a single queue and a fixed number of servers were considered. For the studied multiple queue problem with unfixed number of servers, the main challenges are:

- Different splitting strategies result in different response time and further lead to different profits. Therefore, how to split the request stream is challenging for profit maximization, and
- The number of on-demand servers is crucial for the total rental cost. Deciding the number of on-demand servers to be rented is also challenging for profit maximization.

How to split the coming request stream is crucial for the total profit, i.e., different request stream splitting strategies lead to different performance. We therefore analyze the performance of the request stream splitting first.

3.1 Performance of Request Stream Splitting

Since requests are processed on either reserved or on-demand servers, they have different impacts on performance. Generally, more requests to be processed on reserved servers or less requests to be processed on on-demand servers in a time unit result in a higher total profit, i.e., it is desirable to minimize the expected response time. (1) *Requests on reserved servers*. For λ_i requests with impatient consumers, an $M/M/n_i^r/n_i^r + R$ queue model is constructed. In queue Q_i, there are $n_i^r + R + 1$ states $\{0, 1, \ldots, n_i^r + R\}$ and each state represents the number of requests. We define a variable $h(x) = 1$ if $x > 0$ and $h(x) = 0$ otherwise. p_j^i $(i \in \{1, \ldots, S\}; j \in \{0, \ldots, n_i^r + R\})$ denotes the steady state probability of Q_i with j requests. Therefore,

$$\sum_{j=0}^{n_i^r+R} p_j^i = 1. \tag{2}$$

According to [35], p_j^i can be calculated using

$$\begin{aligned} p_j^i = {} & h(n_i^r + 1 - j) p_0^i \frac{\lambda_i^j}{\mu_i^j j!} \\ & + h(j - n_i^r) p_0^i \frac{\lambda_i^j}{\mu_i^{n_i^r - 1} (n_i^r - 1)! \prod_{k=0}^{j-n_i^r} (n_i^r \mu_i + k\theta)}. \end{aligned} \tag{3}$$

Let

$$q_{i,j}^+ = \frac{\lambda_i^j}{\mu_i^{n_i^r - 1} (n_i^r - 1)! \prod_{k=0}^{j-n_i^r} (n_i^r \mu_i + k\theta)}$$

for $j \in \{n_i^r + 1, \ldots, n_i^r + R\}$, and $q_{i,j}^- = \frac{\lambda_i^j}{\mu_i^j j!}$ for $j \in \{0, 1, \ldots, n_i^r\}$. According to Equations (2) and (3), p_0^i can be obtained by

$$p_0^i = \frac{1}{\sum_{j=0}^{n_i^r} q_{i,j}^- + \sum_{j=n_i^r+1}^{n_i^r+R} q_{i,j}^+}. \tag{4}$$

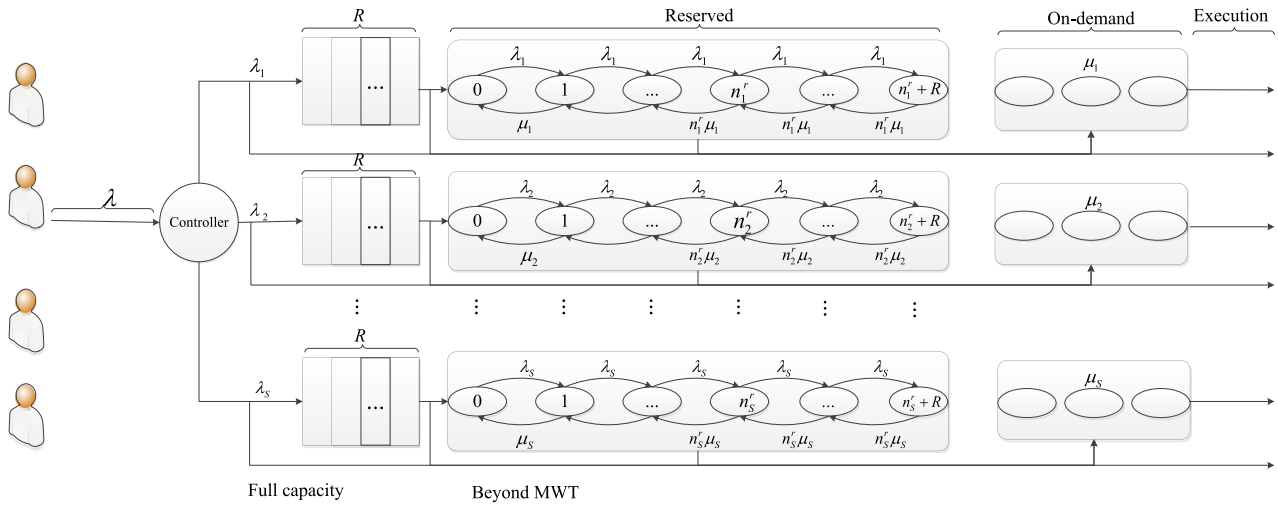


Fig. 1. The system queue model.

Therefore, the expected number of requests $L(\lambda_i)$ of Q_i in the queue is

$$L(\lambda_i) = \sum_{j=n_i^r+1}^{n_i^r+R} (j - n_i^r) \cdot p_j^i. \tag{5}$$

Note that $L(\lambda_i)$ includes the requests waiting in Q_i. The expected waiting time $\overline{W}_i$ is $\frac{L(\lambda_i)}{\lambda_i}$ in the system. Therefore, $\overline{W} = \sum_{i=1}^{S} \frac{\lambda_i}{\lambda} \overline{W}_i = \frac{1}{\lambda} \sum_{i=1}^{S} L(\lambda_i)$. For the S types of reserved servers, the optimal split of λ into $(\lambda_1^*, \ldots, \lambda_S^*)$ satisfies

$$\overline{W}^* = \min\{\overline{W}\} = \frac{1}{\lambda} \min\Big\{ \sum_{i=1}^{S} L(\lambda_i) \Big\}. \tag{6}$$

(2) *Requests on on-demand servers*. Let W_i^j be the waiting time of a new request coming to Q_i with j requests. According to [35], the probability of $W_i^j \leq \frac{1}{\theta}$ is

$$P\left(W_i^j \leq \frac{1}{\theta}\right) = \frac{n_i^r \mu_i}{n_i^r \mu_i + h(j - n_i^r + 1)(j - n_i^r)\theta}. \tag{7}$$

The probability of a new request exceeding the MWT P_M^i can be calculated by

$$\begin{aligned} P_M^i &= \sum_{j=n_i^r}^{n_i^r+R-1} p_j^i P\left(W_i^j > \frac{1}{\theta}\right) \\ &= \sum_{j=n_i^r}^{n_i^r+R-1} p_j^i \left(1 - P\left(W_i^j \leq \frac{1}{\theta}\right)\right). \end{aligned} \tag{8}$$

According to the above two cases for on-demand request allocation, the probability P_i^o of a request being processed on an on-demand server in Q_i is

$$P_i^o = P_M^i + p_{n_i^r+R}^i. \tag{9}$$

Therefore, the number of requests being processed on on-demand servers is $\sum_{i=1}^{S} \lambda_i P_i^o$, which is also closely related to the split strategy, i.e., it is desirable to have an optimal split $(\lambda_1^*, \ldots, \lambda_S^*)$.

3.2 Mathematical Model for the Problem

With respect to the expected response time $\overline{T}$, the optimal split $(\lambda_1^*, \ldots, \lambda_S^*)$ can be obtained. However, the total profit of the arriving requests per unit time depends on the strategy for actually allocating them to the reserved or on-demand servers. After the requests in the coming stream are split into different queues, the actual response time T_i^{os} of requests in Q_i is determined by the waiting time in the queue and the processing time on servers. During the renting period ω, $\lceil \frac{\omega}{T_i^{os}} \rceil$ time units are required. The ratio of processing time of each request to a time unit is $\lceil \frac{1}{\mu_i} \rceil$. Different allocation strategies lead to different total profits. More specifically, the total profit β is closely related to the rental cost of the reserved servers $n_i^r c_i^r \omega$ $(i \in \{1, 2, \ldots, S\})$ and that of on-demand servers $n_i^o c_i^o \lceil \frac{\omega}{T_i^{os}} \rceil \lceil \frac{1}{\mu_i} \rceil$ $(i \in \{1, 2, \ldots, S\})$. The considered problem can be mathematically modeled as follows:

$$\max \beta = \sum_{i=1}^{S} \left(\lambda_i \alpha \lceil \frac{\omega}{T_i^{os}} \rceil - \left(n_i^r c_i^r \omega + n_i^o c_i^o \lceil \frac{\omega}{T_i^{os}} \rceil \lceil \frac{1}{\mu_i} \rceil \right) \right) \tag{10}$$

$$s.t. : \sum_{i=1}^{S} \lambda_i = \lambda \tag{11}$$

$$n_i^r \mu_i \geq \lambda_i (1 - P_i^o), \forall i \in \{1, 2, \ldots, S\} \tag{12}$$

$$\lambda \geq \lambda_i \geq 0, \forall i \in \{1, 2, \ldots, S\} \tag{13}$$

$$n_i^r \geq 0, \forall i \in \{1, 2, \ldots, S\} \tag{14}$$

$$n_i^o \mu_i \geq \lambda_i P_i^o, \forall i \in \{1, 2, \ldots, S\} \tag{15}$$

$$n_i^o \geq 0, \forall i \in \{1, 2, \ldots, S\} \tag{16}$$

$$c_i^r > 0, \forall i \in \{1, 2, \ldots, S\} \tag{17}$$

$$c_i^o > 0, \forall i \in \{1, 2, \ldots, S\} \tag{18}$$

$$\alpha > \frac{c_S^o P_i^o}{\mu_i^2}, \forall i \in \{1, 2, \ldots, S\}. \tag{19}$$

Equation (10) indicates that the total profit is determined by both the total income $\sum_{i=1}^{S} \lambda_i \alpha \lceil \frac{\omega}{T_i^{os}} \rceil$ and the total cost $\sum_{i=1}^{S} (n_i^r c_i^r \omega + n_i^o c_i^o \lceil \frac{\omega}{T_i^{os}} \rceil \lceil \frac{1}{\mu_i} \rceil)$. The total income is affected by T_i^{os} and λ_i while the total cost is influenced by both the number of rented servers and T_i^{os}. For the optimal split $(\lambda_1^*, \ldots, \lambda_S^*)$, β depends only on $(T_1^{os}, \ldots, T_S^{os})$ in terms of Theorem 1.

Theorem 1. *The profit of Q_i ($i \in \{1, \ldots, S\}$) with λ_i^* is inversely proportional to the actual response time T_i^{os}.*

Proof. Equation (10) implies that the profit of Q_i ($i \in \{1, \ldots, S\}$) with λ_i^* is $(\lambda_i^* \alpha - \frac{n_i^o c_i^o}{\mu_i}) \frac{\omega}{T_i^{os}} - n_i^r c_i^r \omega$ in which $n_i^r c_i^r \omega$ is a constant. According to Equation (15), $n_i^o = \frac{\lambda_i^* P_i^o}{\mu_i}$ with the minimal rental cost of on-demand servers. Since $c_i^o \leq c_S^o$ and $\alpha > \frac{c_S^o P_i^o}{\mu_i^2}$, $\lambda_i^* \alpha - \frac{n_i^o c_i^o}{\mu_i} = \lambda_i^* (\alpha - \frac{c_i^o P_i^o}{\mu_i^2}) \geq \lambda_i^* (\alpha - \frac{c_S^o P_i^o}{\mu_i^2}) \geq 0$. Therefore, the profit of Q_i is inversely proportional to T_i^{os}. □

4 The Proposed Methods

According to Equation (10) and Theorem 1, the profit β is affected by $\lambda_i, T_i^{os}, n_i^r$ and n_i^o ($i \in \{1, \ldots, S\}$). To maximize the total profit, an optimal n_i^r and n_i^o should be provisioned to Q_i. According to Equation (6), it is desirable to optimally split λ to minimize the expected response time, which implies that most requests might be processed with the maximum profit. Furthermore, T_i^{os} is minimized by optimally allocating the requests. Consequently, there are three optimization subproblems in the considered problem:

(i) Optimally splitting λ into $(\lambda_1^*, \ldots, \lambda_S^*)$ for the given number of reserved servers n_i^r and the corresponding on-demand servers n_i^o ($i \in \{1, \ldots, S\}$),
(ii) Optimally allocating requests of each queue to servers so as to minimize T_i^{os} ($i \in \{1, \ldots, S\}$), and
(iii) Repeatedly searching to obtain the optimal number of servers n_i^r ($i \in \{1, \ldots, S\}$).

Due to the considered impatience and hybrid resource provision, there are new challenges in these subproblems as compared to the existing studies [31], [36]. All these three subproblems are NP-hard, meaning that the considered problem is also NP-hard.

To deal with the considered problem, an MQRS (Multi-Queue Request Scheduling) algorithm framework is proposed as depicted in Algorithm 1. The optimal numbers of reserved servers $\vec{n}^{*r}$ are initialized to $\left(\lceil \frac{\lambda}{m\mu_1} \rceil, \ldots, \lceil \frac{\lambda}{m\mu_S} \rceil\right)$. Here $\vec{n}^r$ is an S-dimensional vector representing the optimal numbers of reserved servers for all the queues in the previous loop. Assume that there are d neighborhood structures [42] in MQRS. $U = (\mu_1, \mu_2, \ldots, \mu_S)$ denotes the service rates for the S queues.

There are three components in MQRS: *Request Stream Splitting* (*RSS*), *Request Allocation* (*RA*), and *Server Number Adjustment* (*SNA*). RSS optimally splits the arrival rate λ into $(\lambda_1^*, \ldots, \lambda_S^*)$ for the given number of servers n_i^r using the Lagrange Multiplier method [37]. According to Theorem 1, RA schedules the requests in each queue to the corresponding servers leading to the minimum actual response time by the one-step improvement policy strategy for semi-Markov decision processes [36]. SNA adjusts and searches the appropriate number of reserved servers. MQRS is repeated until $\vec{n}^r = \vec{n}^{*r}$. The computational time complexity of MQRS is hard to estimate because of the number of iterations is not fixed. However, in the worst case, it can be estimated as $O\Big(\lceil \frac{\lambda}{m\mu_1} \rceil^S d^2 S \max\{\frac{\lambda}{m\epsilon}, S^2(\lceil \frac{\lambda}{m\mu_1} \rceil + R)\}\Big)$.

Algorithm 1. Multi-Queue Request Scheduling (MQRS)

```
1  begin
2    β ← 0, n⃗^r ← 0⃗;
3    for i = 1 to S do
4      n_i^{*r} ← ⌈λ/(mμ_i)⌉;
5    while n⃗^r ≠ n⃗^{*r} do
6      k ← 0, t ← 0, n⃗^r ← n⃗^{*r};
7      while k ≤ d do
8        β' ← 0, n⃗_1^r ← n⃗^{*r};
9        for k_1 = 1 to S do
10         for k_2 = 0 to 2k do
11           Call RSS;// Request Stream Splitting (Algorithm 3)
12           Call RA;// Request Allocation (Algorithm 5)
13           Compute n⃗^{r'} and β' by calling SNA;
             // Adjustment the number of servers (Algorithm 6)
14           if (n_{k_1}^{r'} < 0 OR n⃗^{r'} U^⊤ < λ) then
15             Continue;
16           if β' ≥ β then
17             n⃗^{*r} ← n⃗^{r'}, β ← β';
18       k ← k + 1;
19     t ← t + 1;
20   return β.
```

4.1 Request Stream Splitting

Since the impatience constraint has a clear influence on the response time that is further closely related to the number of requests in each queue, it is essential to appropriately split the request stream. Equation (6) implies that minimizing $\overline{T}$ is related to minimizing the number of requests $\sum_{i=1}^{S} L(\lambda_i)$ in the queues. The request stream splitting problem can be modeled mathematically as follows:

$$\min \sum_{i=1}^{S} L(\lambda_i) \tag{20}$$

$$s.t. \sum_{i=1}^{S} \lambda_i = \lambda \tag{21}$$

$$\lambda_i - n_i^r\mu_i \le 0, \forall i \in \{1,2,\ldots,S\} \tag{22}$$

$$\lambda_i \ge 0, \forall i \in \{1,2,\ldots,S\}. \tag{23}$$

Let $\tilde{\lambda} = (\lambda_1,\ldots,\lambda_S)$ be a splitting vector of the arrival rate of requests to S queues, $\vec{\zeta}$ and $\vec{\upsilon}$ be $S-$dimensional vectors, and ϕ a Lagrange multiplier. We observe that $\phi\Big(\sum_{i=1}^{S}\lambda_i - \lambda\Big) = 0$, $\sum_{i=1}^{S}\vec{\zeta}_i\lambda_i = 0$ and $\sum_{i=1}^{S}\vec{\upsilon}_i(n_i^r\mu_i - \lambda_i) = 0$. According to the Lagrange Multiplier method [37], the objective function can be equivalently transformed into

$$\begin{aligned}\min H(\tilde{\lambda},\phi,\vec{\zeta},\vec{\upsilon}) = &\sum_{i=1}^{S}L(\lambda_i) - \phi\Big(\sum_{i=1}^{S}\lambda_i - \lambda\Big)\\ &- \sum_{i=1}^{S}\vec{\zeta}_i\lambda_i - \sum_{i=1}^{S}\vec{\upsilon}_i(n_i^r\mu_i - \lambda_i).\end{aligned} \tag{24}$$

The Karush-Kuhn-Tucher (KKT) conditions [38] of Equation (24) satisfy:

$$\begin{cases}\frac{\partial L(\lambda_i)}{\partial\lambda_i} - \phi - \vec{\upsilon}_i - \vec{\zeta}_i = 0, \ (i \in \{1,\ldots,S\}) & (25)\\ \vec{\zeta}_i\lambda_i = 0, \ (i \in \{1,\ldots,S\}) & (26)\\ \vec{\upsilon}_i(n_i^r\mu_i - \lambda_i) = 0, \ (i \in \{1,\ldots,S\}) & (27)\\ \sum_{i=1}^{S}\lambda_i - \lambda = 0. & (28)\end{cases}$$

Equation (26) implies that: (i) $\vec{\zeta}_i = 0$ if $\lambda_i > 0$, and (ii) the i^{th} type server is not rented if $\lambda_i = 0$, i.e., $\vec{\zeta}_i$ can be set as 0. Similarly, Equation (27) indicates that $\vec{\upsilon}_i = 0$ if $\lambda_i < n_i^r\mu_i$ and $\vec{\upsilon}_i$ can be set as 0 when $\lambda_i = n_i^r\mu_i$. Therefore, only the following conditions are necessary:

$$\begin{cases}\frac{\partial L(\lambda_i)}{\partial\lambda_i} - \phi = 0, \ (i \in \{1,\ldots,S\}) & (29)\\ \sum_{i=1}^{S}\lambda_i - \lambda = 0 & (30)\end{cases}$$

Theorem 2. *$L(\lambda_i)$ is non-decreasing with the increase of λ_i ($\lambda_i \in (0, n_i^r\mu_i]$).*

Proof. Equation (3) implies that $\frac{\partial p_j^i}{\partial\lambda_i}$ ($i \in \{1,\ldots,S\}$) is determined by $\frac{\partial p_0^i}{\partial\lambda_i}$. From Equation (4) and

$$q_{i,j-1}^{+} = \frac{\lambda_i^{j-1}}{\mu_i^{n_i^r-1}(n_i^r-1)!\prod_{k=0}^{j-1-n_i^r}(n_i^r\mu_i + k\theta)},$$

we have

$$\begin{aligned}\frac{\partial p_0^i}{\partial\lambda_i} &= -\frac{\sum_{j=1}^{n_i^r}\frac{q_{i,(j-1)}^{-}}{\mu_i} + \sum_{j=n_i^r+1}^{n_i^r+R}\frac{jq_{i,(j-1)}^{+}}{n_i^r\mu_i+(j-n_i^r)\theta}}{\Big(\sum_{j=0}^{n_i^r-1}q_{i,j}^{-} + \sum_{j=n_i^r}^{n_i^r+R}q_{i,j}^{+}\Big)^2}\\ &= -(p_0^i)^2\left(\sum_{j=1}^{n_i^r}\frac{q_{i,(j-1)}^{-}}{\mu_i} + \sum_{j=n_i^r+1}^{n_i^r+R}\frac{jq_{i,(j-1)}^{+}}{n_i^r\mu_i+(j-n_i^r)\theta}\right)\\ &= -\frac{(p_0^i)^2(1-p_0^i)j}{\lambda_i}.\end{aligned}$$

According to Equation (5), $\frac{\partial L(\lambda_i)}{\partial\lambda_i} = \sum_{j=n_i^r+1}^{n_i^r+R}(j-n_i^r)\cdot\frac{\partial p_j^i}{\partial\lambda_i}$. Therefore,

$$\begin{aligned}\phi = \frac{\partial L(\lambda_i)}{\partial\lambda_i} &= \sum_{j=n_i^r+1}^{n_i^r+R}(j-n_i^r)\Big(p_0^i\frac{jq_{i,(j-1)}^{+}}{n_i^r\mu_i+(j-n_i^r)\theta} + \frac{\partial p_0^i}{\partial\lambda_i}p_j^i\Big)\\ &= \sum_{j=n_i^r+1}^{n_i^r+R}(j-n_i^r)p_0^i\frac{jq_{i,(j-1)}^{+}}{n_i^r\mu_i+(j-n_i^r)\theta}\Big(1-p_0^i+(p_0^i)^2\Big)\\ &= \sum_{j=n_i^r+1}^{n_i^r+R}(j-n_i^r)p_0^i\frac{jq_{i,(j-1)}^{+}}{n_i^r\mu_i+(j-n_i^r)\theta}\Big((p_0^i-\frac{1}{2})^2+\frac{3}{4}\Big).\end{aligned} \tag{31}$$

Therefore, $\phi \ge 0$, which means that $\frac{\partial L(\lambda_i)}{\partial\lambda_i} \ge 0$ and $L(\lambda_i)$ is non-decreasing with the increase of λ_i ($\lambda_i \in (0, n_i^r\mu_i]$). □

Theorem 3. *ϕ is non-decreasing with the increase of λ_i ($\lambda_i \in (0, n_i^r\mu_i]$).*

Proof. Let $\xi = (j-n_i^r)p_0^i\frac{jq_{i,(j-2)}^{+}}{(n_i^r\mu_i+(j-n_i^r)\theta)(n_i^r\mu_i+(j-1-n_i^r)\theta)} > 0$. From Equation (31), we have

$$\begin{aligned}\frac{\partial\phi}{\partial\lambda_i} &= \sum_{j=n_i^r+1}^{n_i^r+R}\xi\Big((j-1)(1-p_0^i+(p_0^i)^2)\\ &\quad -(1-2p_0^i+3(p_0^i)^2)(p_0^i-(p_0^i)^2)j\Big)\\ &= \sum_{j=n_i^r+1}^{n_i^r+R}\xi\Big((j-1)(1-p_0^i+(p_0^i)^2)\\ &\quad -(1-2p_0^i+3(p_0^i)^2)(p_0^i-(p_0^i)^2)j\Big)\\ &= \sum_{j=n_i^r+1}^{n_i^r+R}\xi\Big(j(3(p_0^i)^4-5(p_0^i)^3+4(p_0^i)^2-2p_0^i+1)\\ &\quad -((p_0^i)^2-p_0^i+1)\Big)\\ &\ge \sum_{j=n_i^r+1}^{n_i^r+R}\xi\Big(j(3(p_0^i)^4-5(p_0^i)^3+4(p_0^i)^2-2p_0^i+1)-1\Big).\end{aligned} \tag{32}$$

Let $\xi_1 = j(3(p_0^i)^4-5(p_0^i)^3+4(p_0^i)^2-2p_0^i+1)$. We obtain $\frac{\partial\xi_1}{\partial p_0^i} = j(12(p_0^i)^3-15(p_0^i)^2+8p_0^i-2)$. When $p_0^i = 0$, $\frac{\partial\xi_1}{\partial p_0^i} < 0$ while when $p_0^i = 1$, $\frac{\partial\xi_1}{\partial p_0^i} > 0$. There is a value $p_0^i \in (0,1)$, $\frac{\partial\xi_1}{\partial p_0^i} = 0$ where $p_0^i = 0.6027$. Therefore, ξ_1 decreases when $p_0^i < 0.6027$ while it increases when $p_0^i > 0.6027$. When $p_0^i = 0.6027$, ξ_1 obtains the minimum value $\xi_1 = 0.5488j$. When the server type is selected, there is at least one server which implies that $j \ge 2$. Therefore, $\xi_1 \ge 0$ which implies that $\frac{\partial\phi}{\partial\lambda_i} \ge 0$. ϕ is non-decreasing with the increase of λ_i ($\lambda_i \in (0, n_i^r\mu_i]$). □

Theorem 2 illustrates that a smaller λ_i ($i \in \{1,\ldots,S\}$) results in a smaller $L(\lambda_i)$ while they are constrained by Equation (30). Therefore, it is crucial to obtain the best estimator $(\lambda_1^*,\ldots,\lambda_S^*)$ with a minimum $\sum_{i=1}^{S}L(\lambda_i^*)$. Since $p_0^i \in [0,1]$, the maximum value of ϕ is $n_i^r\mu_i$ for each queue. Therefore, the upper bound of ϕ is $\phi_{\max} = \min_{i\in\{1,\ldots,S\}}\{n_i^r\mu_i\}$, i.e., $\phi \in [0,\phi_{\max}]$. Equation (31) means that ϕ is determined by p_0^i. Both of them depend on λ_i. The RSS (Request Stream Splitting) algorithm obtains λ_i^* ($i \in \{1,\ldots,S\}$) by changing ϕ from 0 to $\phi_{\max}$. For each ϕ, λ_i

is iteratively estimated by the binary search method as depicted in Algorithm 2. Since $\Lambda = \sum_{i=1}^{S} \lambda_i$ might be more or less than λ, λ_i is normalized by $\frac{\lambda\lambda_i}{\Lambda}$, which is denoted by λ_i^*. The best estimator $(\lambda_1^*, \ldots, \lambda_S^*)$ obtains the minimum $\sum_{i=1}^{S} L(\tilde{\lambda}_i)$ in terms of Theorem 3. RSS is described in Algorithm 3.

For a given ϕ, λ_i is iteratively searched by the following binary search procedure. The lower bound $\lambda_{\min}$ and the upper bound $\lambda_{\max}$ of requests are initialized to 0.001 and $n_i^r\mu_i$ respectively. Theorem 3 demonstrates that ϕ increases with an increase of λ_i. For $\lambda_i = \frac{\lambda_{\min}+\lambda_{\max}}{2}$, p_0^i and ϕ_1 are calculated by Equations (4) and (31) respectively. If $\phi_1 \geq \phi$, ϕ increases with the increase of λ_i, which implies that λ_i locates in $[\lambda_{\min}, \frac{\lambda_{\max}+\lambda_{\min}}{2}]$, i.e., $\lambda_{\max} \leftarrow \frac{\lambda_{\max}+\lambda_{\min}}{2}$ and $\lambda_{\min} \leftarrow \frac{\lambda_{\max}+\lambda_{\min}}{2}$ otherwise. The procedure terminates when $\lambda_{\max} - \lambda_{\min} > \epsilon$.

Algorithm 2. Arrival Rate Estimation (ARE)

Input: $n_i^r, \mu_i, R, \phi, \epsilon$
begin
  $\lambda_{\min} \leftarrow 0.001$, $\lambda_{\max} \leftarrow n_i^r\mu_i$;
  while $\lambda_{\max} - \lambda_{\min} > \epsilon$ **do**
    $\lambda_i \leftarrow \frac{\lambda_{\max}+\lambda_{\min}}{2}$;
    Calculate ϕ_1 by Equation (31);
    if $\phi_1 \geq \phi$ **then**
      $\lambda_{\max} \leftarrow \lambda_i$;
    else
      $\lambda_{\min} \leftarrow \lambda_i$;
  return λ_i.

To illustrate the procedure of Algorithm 2, the following example is given: $\lambda = 3$, $R = 2$, $\theta = 10$, $n_i^r = 3$, $\epsilon = 0.1$, $\mu_i = 0.6$, $\phi = 0.0494$. According to the given parameters, the values calculated by Algorithm 2 are shown in Table 2. The time complexity of Algorithm 2 is $O(\log(\frac{n_i^r\mu_i}{\epsilon}))$.

Algorithm 3. Request Stream Splitting (RSS) Algorithm

Input: $\lambda, \mu_1, \ldots, \mu_S, \vec{n}^r, R, Z$
begin
  $\Lambda \leftarrow 0$, $\phi_{\max} \leftarrow 1000$, $\phi_{\min} \leftarrow 0$;
  for $i = 1$ **to** S **do**
    $\lambda_i^* \leftarrow n_i^r\mu_i$, $\Lambda \leftarrow \Lambda + \lambda_i^*$;
    Calculate ϕ in terms of Equation (31);
    if $\phi_{\max} > \phi$ **then**
      $\phi_{\max} \leftarrow \phi$;
  while $\phi_{\max} - \phi_{\min} \leq 0.001$ **do**
    $\phi \leftarrow \frac{\phi_{\min}+\phi_{\max}}{2}$;
    $\Lambda \leftarrow 0$, $L \leftarrow 0$, $F \leftarrow 0$;
    for $i = 1$ **to** S **do**
      Calculate λ_i by Algorithm 2;
      $\tilde{\lambda}_i \leftarrow \lambda_i$, $\Lambda \leftarrow \Lambda + \lambda_i$;
    if $\Lambda < \lambda$ **then**
      $\phi_{\min} \leftarrow \phi$;
    else
      $\phi_{\max} \leftarrow \phi$;
  for $i = 1$ **to** S **do**
    $\lambda_i^* \leftarrow \frac{\lambda}{\Lambda}\tilde{\lambda}_i$;
  return $(\lambda_1^*, \ldots, \lambda_S^*)$.

TABLE 2
An Example on Arrival Rate Estimation

$\lambda_{\min}$	$\lambda_{\max}$	λ_i	ϕ_1
0.001	1.80	0.90	0.0408
0.90	1.80	1.35	0.0835
0.90	1.35	1.1254	0.0618
0.90	1.1254	1.0129	0.0511
0.90	1.0129	0.9567	0.0458
0.9567	1.0129	0.9848	0.0484
0.9848	1.0129	0.9989	0.0497
0.9848	0.9989	0.9919	0.0491

To illustrate the process of Algorithm 3, the same example for Algorithm 2 is used with $S = 2$, $\vec{n}^r = (3, 2)$, $\mu_1 = 0.6$, $\mu_2 = 0.8$. The results are shown in Table 3 and the final arrival rate splitting vector is (1.46,1.54). The time complexity of Algorithm 3 is $O(\max(S, \log(1000\phi_{\max})))$.

4.2 Request Allocation

Theorem 1 illustrates that a smaller response time T_i^{os} leads to a higher profit of Q_i ($i \in \{1, \ldots, S\}$) with λ_i^*. To maximize the total profit, it is desirable to obtain the minimum T_i^{os} of each queue Q_i ($i \in \{1, \ldots, S\}$). RSS obtains the optimal request split, i.e., how to optimally split requests. However, which requests should be allocated to each Q_i ($i \in \{1, \ldots, S\}$) has a great impact on T_i^{os}. Besides the newly arriving requests, there are some requests in Q_i waiting to be processed. Since the processing time of each request is given, T_i^{os} depends on the waiting time that is affected by the number of requests in Q_i. Therefore, minimizing T_i^{os} is related to minimizing the number of requests in Q_i, i.e., maximizing the total profit implies minimizing the number of requests in the system.

For each queue Q_i with arrival rate λ_i and n_i^r servers, the ratio $\frac{\lambda_i^*}{\lambda}$ ($i \in \{1, \ldots, S\}$) indicates the probability of a new request to be allocated to Q_i. The Poisson process of arriving requests means that the arriving requests allocated to each queue also follow a Poisson distribution with rate λ_i^*. Therefore, each queue system can be constructed using an $M/M/n_i^r/n_i^r + R$ model [1]. $\vec{\mathbf{x}} = (\vec{x}_1, \ldots, \vec{x}_S)$ is represented as a system state where $\vec{x}_i \in \{0, 1, \ldots, n_i^r + R\}$ ($i \in \{1, \ldots, S\}$) is the number of requests in Q_i. When a new request arrives at the system, the Controller allocates the request to a certain queue using the following strategy.

Let $a = i$ ($i \in \{1, \ldots, S\}$) be an action which means that the new request is assigned to Q_i. The request allocation procedure is a semi-Markov decision problem that makes

TABLE 3
An Example on Request Stream Splitting

$\phi_{\min}$	$\phi_{\max}$	ϕ	$\tilde{\lambda}$
0	0.0987	0.0494	(1.5777,1.4223)
0.0494	0.0987	0.0740	(1.5165,1.4835)
0.0740	0.0987	0.0864	(1.4887,1.5113)
0.0864	0.0987	0.0925	(1.4671,1.5239)
0.0925	0.0987	0.0956	(1.4642,1.5358)
0.0925	0.0956	0.0941	(1.4620,1.5380)
0.0925	0.0941	0.0949	(1.4631,1.5369)

the allocation decision only when a new request arrives. For an initial state $\vec{\mathbf{x}}$, the new request takes either the action a or the $(\frac{\lambda_1^*}{\lambda},\ldots,\frac{\lambda_S^*}{\lambda})$ allocation policy. $\Delta\Big(\vec{\mathbf{x}},a,(\frac{\lambda_1^*}{\lambda},\ldots,\frac{\lambda_S^*}{\lambda})\Big)$ is the difference between them, which is defined as the total expected number of requests in the system over an infinitely long period of time. According to [36], minimizing the number of requests in the system is equivalent to minimizing $\Delta\Big(\vec{\mathbf{x}},a,(\frac{\lambda_1^*}{\lambda},\ldots,\frac{\lambda_S^*}{\lambda})\Big)$ for each state $\vec{\mathbf{x}}$. The semi-Markov decision problem can be solved by the one-step policy improvement (OSPI) procedure that consists of two steps [39]: (i) calculating $(\frac{\lambda_1^*}{\lambda},\ldots,\frac{\lambda_S^*}{\lambda})$ and (ii) computing $\Delta\Big(\vec{\mathbf{x}},a,(\frac{\lambda_1^*}{\lambda},\ \ldots,\frac{\lambda_S^*}{\lambda})\Big)$ by the OSPI procedure.

Let $v_i(j)$ represent the difference of the number of expected requests in Q_i between starting from $j+1$ requests and from j requests over an infinitely long period of time. $\Delta\Big(\vec{\mathbf{x}},a,(\frac{\lambda_1^*}{\lambda},\ldots,\frac{\lambda_S^*}{\lambda})\Big)$ of each state $\vec{\mathbf{x}}$ with action $a=i$ can be calculated by

$$\Delta\left(\vec{\mathbf{x}},a,\left(\frac{\lambda_1^*}{\lambda},\ldots,\frac{\lambda_S^*}{\lambda}\right)\right)=v_i(\vec{x}_i)-\sum_{j=1}^{S}\frac{\lambda_j^*}{\lambda}v_j(\vec{x}_j). \tag{33}$$

Since $\sum_{j=1}^{S}\frac{\lambda_j^*}{\lambda}v_j(\vec{x}_j)$ is independent to the action $a=i$, it is desirable to allocate the new request to Q_i to obtain $\min_{i\in\{1,\ldots,S\}}v_i(\vec{x}_i)$.

During the dynamic request coming/departure process from the $\vec{x}_i$ state first returning to the empty state of Q_i, the total expected number of requests is $K_i(\vec{x}_i)$ and the expected time is $T_i(\vec{x}_i)$. By defining

$$w_i(\vec{x}_i)=K_i(\vec{x}_i)-L_i(\lambda_i^*)T_i(\vec{x}_i), \tag{34}$$

we can calculate $v_i(\vec{x}_i)$ using

$$v_i(\vec{x}_i)=w_i(\vec{x}_i+1)-w_i(\vec{x}_i). \tag{35}$$

$K_i(\vec{x}_i)$ is obtained according to [36] as

$$\begin{aligned}K_i(\vec{x}_i)=&\Big[\frac{\vec{x}_i}{\lambda_i^*+\vec{x}_i\mu_i}+\frac{\vec{x}_i\mu_i}{\lambda_i^*+\vec{x}_i\mu_i}K_i(\vec{x}_i-1)\\&+\frac{\lambda_i^*}{\lambda_i^*+\vec{x}_i\mu_i}K_i(\vec{x}_i+1)\Big]h(n_i^r+1-\vec{x}_i)\\&+\Big\{\frac{\vec{x}_i-n_i^r}{n_i^r\mu_i-\lambda_i^*}\Big[\frac{1}{2}(\vec{x}_i-n_i^r+1)++n_i^r\\&+\frac{\lambda_i^*}{n_i^r\mu_i-\lambda_i^*}\Big]\Big\}h(\vec{x}_i-n_i^r).\end{aligned} \tag{36}$$

There are n_i^r+R+1 states corresponding to the number of requests in Q_i. Since $K_i(0)=0$, there are n_i+R unknown quantities $K_i(\vec{x}_i)$ $(\vec{x}_i\in\{1,2,\ldots,n_i\})$ which can be solved by the n_i^r equations determined by Equation (36). Let $\vec{K}_i=(K_i(1),\ldots,K_i(n_i^r+R))$. The n_i^r equations can be represented as $\mathbf{A}_i\vec{K}_i^{\top}=\vec{B}_i$ where $\mathbf{A}_i$ is

$$\mathbf{A}_i=\begin{pmatrix}1&\frac{-\lambda_i^*}{\lambda_i^*+\mu_i}&&\\\frac{-2\mu_i}{\lambda_i^*+2\mu_i}&1&\frac{-\lambda_i^*}{\lambda_i^*+2\mu_i}&\\&\ddots&\ddots&\\&&\frac{-n_i^r\mu_i}{\lambda_i^*+n_i^r\mu_i}&1\end{pmatrix},$$

and $\vec{B}$ is

$$\vec{B}=\left(\frac{1}{\lambda_i^*+\mu_i},\frac{2}{\lambda_i^*+\mu_i},\ldots,\frac{n_i^r}{\lambda_i^*+n_i^r\mu_i}\right).$$

According to [36], $T_i(\vec{x}_i)$ is calculated by

$$\begin{aligned}T_i(\vec{x}_i)=&\Big[\frac{1}{\lambda_i^*+\vec{x}_i\mu_i}+\frac{\vec{x}_i\mu_i}{\lambda_i^*+\vec{x}_i\mu_i}T_i(\vec{x}_i-1)\\&+\frac{\lambda_i^*}{\lambda_i^*+\vec{x}_i\mu_i}T_i(\vec{x}_i+1)\Big]h(n_i^r+1-\vec{x}_i)\\&+\Big[\frac{\vec{x}_i-n_i^r}{n_i^ru_i-\lambda_i^*}+T_i(n_i^r)\Big]h(\vec{x}_i-n_i^r).\end{aligned} \tag{37}$$

Similarly, $T_i(0)=0$ and let $\vec{T}_i=(T_i(1),\ldots,T_i(n_i^r+R))$. The n_i^r equations derived from Equation (37) can be denoted as $\mathbf{A}_i\vec{T}_i^{\top}=\vec{C}_i$ where

$$\vec{C}_i=\left(\frac{1}{\lambda_i^*+\mu_i},\frac{1}{\lambda_i^*+\mu_i},\ldots,\frac{1}{\lambda_i^*+n_i^r\mu_i}\right).$$

Algorithm 4. Elimination Method

Input: $\hat{\lambda},\mu_1,\ldots,\mu_S,\vec{n}^r,R,\theta,\vec{N}$

begin
  $\mathbf{K}\leftarrow\mathbf{0}$, $\mathbf{T}\leftarrow\mathbf{0}$, $\vec{v}\leftarrow\vec{0}$;
  for $i=1$ **to** S **do**
    $\mathbf{A}\leftarrow\mathbf{0}$, $\vec{B},\vec{C}\leftarrow\vec{0}$, $\vec{B}_1\leftarrow\frac{1}{\lambda_i+\mu_i}$, $\vec{C}_1\leftarrow\vec{B}_1$;
    for $j=1$ **to** n_i^r **do**
      $\mathbf{A}_{jj}\leftarrow1$;
      if $1<j<n_i^r$ **then**
        $\mathbf{A}_{j(j-1)}\leftarrow\frac{-j\mu_i}{\lambda_i^*+j\mu_i}$, $\mathbf{A}_{j(j+1)}\leftarrow\frac{-\lambda_i^*}{\lambda_i^*+j\mu_i}$;
        $\vec{C}_j\leftarrow\frac{1}{\lambda_i^*+\mu_i}$, $\vec{B}_j\leftarrow\frac{j}{\lambda_i^*+\mu_i}$;
      else
        $\mathbf{A}_{j(j-1)}\leftarrow\frac{-j\mu_i}{\lambda_i^*+j\mu_i}$;
        $\vec{C}_j\leftarrow\frac{1}{\lambda_i^*+\mu_i}$, $\vec{B}_j\leftarrow\frac{j}{\lambda_i^*+\mu_i}$;
    for $j=2$ **to** n_i^r **do**
      $\mathbf{A}_{jj}\leftarrow\mathbf{A}_{jj}-\frac{\mathbf{A}_{j(j-1)}}{\mathbf{A}_{(j-1)(j-1)}}\mathbf{A}_{(j-1)j}$;
      $\mathbf{A}_{j(j+1)}\leftarrow\mathbf{A}_{j(j+1)}-\frac{\mathbf{A}_{j(j-1)}}{\mathbf{A}_{(j-1)(j-1)}}\mathbf{A}_{(j-1)(j+1)}$;
      $\vec{B}_j\leftarrow\vec{B}_j-\frac{\mathbf{A}_{j(j-1)}}{\mathbf{A}_{(j-1)(j-1)}}\vec{B}_{j-1}$;
      $\vec{C}_j\leftarrow\vec{C}_j-\frac{\mathbf{A}_{j(j-1)}}{\mathbf{A}_{(j-1)(j-1)}}\vec{C}_{j-1}$, $\mathbf{A}_{j(j-1)}\leftarrow0$;
    $\vec{K}_j\leftarrow\frac{\vec{B}_j}{\mathbf{A}_{jj}}$, $\vec{T}_j\leftarrow\frac{\vec{C}_j}{\mathbf{A}_{jj}}$;
    for $j=n_i^r$ **to** 1 **do**
      $\vec{K}_j\leftarrow\frac{\vec{B}_j-\mathbf{A}_{j(j+1)}\vec{K}_{j+1}}{\mathbf{A}_{jj}}$, $\vec{T}_j\leftarrow\frac{\vec{C}_j-\mathbf{A}_{j(j+1)}\vec{T}_{j+1}}{\mathbf{A}_{jj}}$;
    for $j=n_i^r+1$ **to** N_i **do**
      Calculate $\vec{K}_j,\vec{T}_j$ by Equations (36) and (37);
    $\mathbf{K}_i\leftarrow\vec{K}$, $\mathbf{T}_i\leftarrow\vec{T}$;
  return $\mathbf{K},\mathbf{T}$

Let $\vec{N} = (N_1, \ldots, N_S)$ in which $N_i = n_i^r + R$ is the maximal number of requests in Q_i. $\vec{K}_i$ and $\vec{T}_i$ can be calculated by the elimination method as depicted in Algorithm 4. $w_i(\vec{x}_i)$ is computed in terms of Equation (34) and $v_i(\vec{x}_i)$ by Equation (35). The new request is allocated to the queue with index $\arg\min_{i\in\{1,\ldots,S\}} v_i(\vec{x}_i)$ to minimize the actual response time. According to [40], the actual response time of the j^{th} request in Q_i calculated by the one-step policy improvement method is

$$T_{ij}^{os} = h(j+1-n_i^r)\Big((T_i(j) - T_i(j-1) - \frac{1}{\mu_i})p_{n_i^r}^i + \frac{1}{\mu_i}\Big) + h(j-n_i^r)\Big(T_{i(j)} - T_i(j-1) - \frac{1}{\mu_i} - \frac{j-n_i^r}{n_i^r\mu_i - \lambda_i})p_{n_i^r}^i + \frac{1}{\mu_i}\Big).$$

The expected actual response time of the requests in Q_i is $T_i^{os} = \sum_{j=1}^{n_i^r+R} p_j^i T_{ij}^{os}$. If T_i^{os} is less than 1, no more requests are processed because of the constant arrival rate λ. Therefore, T_i^{os} is calculated by

$$T_i^{os} = h(T_i^{os} - 1)\sum_{j=0}^{n_i^r+R} p_j^i T_{ij}^{os} + h(1 - T_i^{os}). \tag{38}$$

The RA (Request Allocation) procedure is described in Algorithm 5. The above example is used to illustrate Algorithm 5. Based on the split arrival rate vector $(1.59, 1.41)$ obtained by Algorithm 3, the results obtained are shown in Table 4. The time complexity of Algorithm 4 is $O(S\max_{i\in\{1,\ldots,S\}}(N_i))$ and that of Algorithm 5 is $O(S^2\max_{i\in\{1,\ldots,S\}}(N_i)) = O\Big(S^2(\max_{i\in\{1,\ldots,S\}}(n_i^r) + R)\Big)$.

Algorithm 5. Request Allocation (RA) Algorithm

Input: $\hat{\lambda}, \mu_1, \ldots, \mu_S, \vec{n}^r, R, \theta, \vec{N}$
begin
  Compute **K**, **T** by calling Algorithm 4;
  for $i = 1$ **to** S **do**
    Calculate L_i using the Equation (5);
    for $j = 1$ **to** N_i **do**
      $\mathbf{w}_{ij} \leftarrow \mathbf{K}_{ij} - L_i\mathbf{T}_{ij}$;
  $\vec{X} \leftarrow \vec{0}, \vec{x} \leftarrow 0, q \leftarrow 1$;
  for $i = 1$ **to** S **do**
    Calculate L_i using the Equation (5);
    for $j = 1$ **to** N_i **do**
      $\mathbf{w}_{ij} \leftarrow \mathbf{K}_{ij} - L_i\mathbf{T}_{ij}$;
  $\mathbf{v}_{i1} \leftarrow \mathbf{w}_{i1}$, $\mathbf{v}_{iN_i} \leftarrow 10$;
  for $j = 2$ **to** $N_i - 1$ **do**
    $\mathbf{v}_{ij} \leftarrow \mathbf{w}_{ij} - \mathbf{w}_{i(j-1)}$;
  while $\vec{x} \leq \vec{N}$ **do**
    for $i = 1$ **to** S **do**
      $\vec{v}_i \leftarrow \mathbf{v}_{i(\vec{x}_i+1)}$;
    $v_{\min} \leftarrow \vec{v}_1$;
    for $i = 2$ **to** S **do**
      if $\vec{v}_i < v_{\min}$ **then**
        $v_{\min} \leftarrow \vec{v}_i, j \leftarrow i$;
    $\vec{x}_j \leftarrow \vec{x}_j + 1, q \leftarrow q + 1$;
  for $i = 1$ **to** S **do**
    Calculate T_i^{os} in terms of Equation (38);
  return T^{os}

TABLE 4
The Request Allocation Example

$\hat{\lambda}$	$\vec{K}$	$\vec{T}$	T^{os}
1.4631	(3.26, 3.91, 3.07, 24.77,52.51)	(1.92, 2.02, 1.42, 2.97,5.94)	1.23
1.5369	(1.27, 1.29, 433.14, 822.11)	(0.96, 0.81, 15.84, 31.68)	2.57

4.3 Adjustment of the Number of Servers

Since the total profit is impacted by the number of servers, it is possible to obtain a better total profit by adjusting the number of servers. The Server Number Adjustment (SNA) algorithm is depicted in Algorithm 6.

The probability P_i^o of requests to be processed by on-demand servers is calculated by Equation (9). The number of on-demand servers is computed by $n_i^o \leftarrow \lceil \frac{\lambda_i^* P_i^o}{\mu_i} \rceil$. The total profit β is calculated by Equation (10). Because different server rates result in various ranges, the neighborhood structure $\mathcal{N}_j^s$ $(j \in \{1, 2, \ldots, d\})$ for $n_i^{r'}$ $(i \in \{1, \ldots, S\})$ of $\vec{n}_1^r$ is changed to $n_i^{r'} \pm j$, where d represents the number of neighborhood structures. It is obvious that $|\mathcal{N}_1^s| \leq |\mathcal{N}_2^s| \leq \ldots \leq |\mathcal{N}_d^s|$ and $|\mathcal{N}_j^s| = 2jS$. Let $\vec{n}^{r'}$ be the neighborhood of $\vec{n}_1^r$, which is determined by the neighborhood structures $\mathcal{N}_j^s$ $(j \in \{1, 2, \ldots, d\})$. The time complexity of Algorithm 6 is $O(S)$.

Algorithm 6. Server Number Adjustment (SNA)

Input: $(\lambda_1^*, \ldots, \lambda_S^*), T^{os}, \vec{n}_1^r, k_1, k_2$
begin
  $\beta' \leftarrow 0$;
  for $i = 1$ **to** S **do**
    Calculate the probability P_i^o by Equation (9);
    $n_i^o \leftarrow \lceil \frac{\lambda_i^* P_i^o}{\mu_i} \rceil$;
    $\beta' \leftarrow \beta' + \lambda_i^*\alpha\lceil\frac{\omega}{T_i^{os}}\rceil - n_i^r c_i^r \omega - n_i^o c_i^o \lceil\frac{\omega}{T_i^{os}}\rceil\lceil\frac{1}{\mu_i}\rceil$;
  $\vec{n}^{r'} \leftarrow \vec{n}_1^r, k_3 \leftarrow (-1)^{k_2}\lfloor\frac{k_2+1}{2}\rfloor, n_{k_1}^{r'} \leftarrow n_{k_1}^{r'} + k_3$;
  return $\vec{n}^{r'}$, β'

To illustrate the process of the overall Multi-Queue Request Scheduling (MQRS, Algorithm 1), the same example for Algorithm 3 is used with ω = 720, c_1^r = 1.59, c_2^r = 2.13, c_1^o = 3.99, c_2^o = 5.31, α = 20, and d=2. Fig. 2 shows the MQRS process for searching the optimal number of servers to maximize the total profit. The optimal solution for each loop is in gray color. According to λ, the initial number of servers is $\vec{n}^{*r}$ = (3,2). If k = 0, the profit for the number of reserved servers $\vec{n}^{*r}$ = (3,2) is estimated. The neighbor set of $\vec{n}^{*r}$ = (3,2) is $\mathcal{N}_1^s$ = {(2,2), (4,2), (3,1), (3,3)} when k = 1.

The arrival rate λ = 3 is split into two rates 1.46 and 1.54 by Algorithm 3. All the requests are allocated to the two queues by Algorithm 5. The profit of each element in $\mathcal{N}_1^s$ is calculated by Algorithm 6. The neighbors (2,2) and (3,1) are not considered because they violate the $\lambda_i^* \leq n_i^r\mu_i$ constraint. By comparing the neighbors, the optimal solution (4,2) is obtained. The neighbor set of (4,2) is obtained $\mathcal{N}_2^s$ ={(4,1), (4,3), (5,2), (3,2), (4,0), (4,4), (2,2), (6,2)} when k = 2 while neighbors (4,0) and (2,2) do not satisfy the $\lambda_i^* \leq n_i^r\mu_i$ constraint. Similarly, the optimal solution (5,2) is obtained. The procedure stops because $\vec{n}^r$ = (5,0) = $\vec{n}^{*r}$.

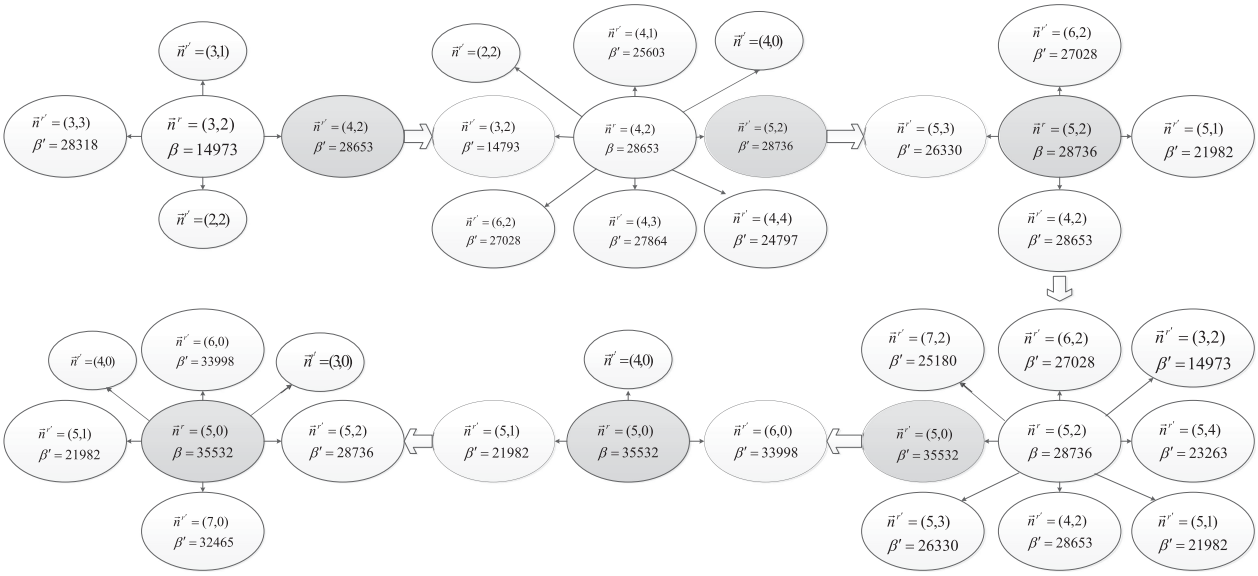


Fig. 2. An illustration of the overall MQRS process.

5 Experimental Evaluation

In the proposed MQRS framework, there are five system parameters and one algorithm component which are calibrated over real life instances. Moreover, the proposed MQRS algorithm is evaluated by comparing it against three similar algorithms over both simulated and real instances. All compared algorithms are coded in Matlab 2017b and run on an Intel Core i7-4770 CPU @3.20 GHz with 8 GBytes of RAM.

5.1 Parameter and Component Calibration

In cloud centers for profit maximization, the commonly tested parameters are: the long term rental period parameter ω, the income per request α, the arrival rate λ, the queue capacity R and the delay rate θ. In [2], most of these parameters are given in advance. To statistically analyze the effects of the system parameters on the proposed algorithm framework, we calibrated these parameters over randomly generated test instances.

Since the values of the calibrated parameters should be as close as possible to real life scenarios, we set the parameter configurations according to the Alibaba cloud[2] and Azure data sets [43] as: $\omega \in${720, 4,320, 8,640} (hours), $\alpha \in${20, 30, 40}, $\lambda \in$[{1, ..., 10}, {11, ..., 20}, {21, ..., 30}, {31, ..., 40}}], $R \in$[1, 6, 10], $\theta \in$[5, 10, 15]. In addition, the total server types $S = 6$ using the general instances by Alibaba cloud: the server rates $\mu \in$[0.1, 0.2, 0.4, 0.6, 0.8, 1.2], the prices for the reserved servers $c^r \in$[0.27, 0.53, 1.06, 1.59, 2.13, 3.19](¥) and the prices for the on-demand servers $c^o \in$[0.66, 1.33, 2.66, 3.99, 5.31, 7.97](¥). Although the server type S is determined, the number of server is dynamic. Furthermore, there is an algorithm parameter which takes values $d \in$ [1, 2, 3, 4]. Therefore, there are $3 \times 3 \times 4 \times 3 \times 3 = 324$ parameter combinations and $1 \times 4 = 4$ component combinations in total. Five instances are generated randomly for each arrival rate λ, i.e., five instances are generated for each combination, that is, $324 \times 4 \times 5 = 6{,}480$ instances in total are tested for calibrating the parameters and the component combinations.

Experimental results are analyzed by the multi-factor analysis of variance (ANOVA) statistical technique. Three main hypotheses (normality, homoscedasticity, and independence of the residuals) are checked from the residuals of the experiments. All three hypotheses can be accepted considering the well-known robustness of the ANOVA technique. The resulting p-values are less than 0.05, meaning that all studied factors have a significant impact on the response variables at the 95 percent confidence level within the ANOVA.

The mean plot of the six studied factors on the total profit β with 95 percent HSD (Tukey's Honest Significance Differences) intervals is shown in Fig. 3 and we can observe that:

(i) ω greatly influences the total profit β. β increases with the increase in ω. The differences are statistically significant. β becomes the minimum when ω = 720. The reason lies in that a higher ω implies a higher profit.

(ii) Similarly, α has a great impact on β and the differences are statistically significant. β increases with an increase in α. β takes the minimum value when $\alpha = 10$ because a bigger α implies more profits.

(iii) λ exerts a great influence on β. With an increase in the upper bound of λ, β increases with statistically significant differences. β is minimum when λ takes values from $\{1, \ldots, 10\}$. The reason lies in that fewer arriving requests translate as expected into few profits.

(iv) R has some minor influence on β. With an increase in R, the influence on β decreases a little. A greater R results in an increase in waiting time of requests.

(v) d has little influence on β with statistically insignificant differences. With an increase in d, the cardinal number of the neighbor set increases. More choices are evaluated to adjust the number of servers. The profit tends to be more stable with an increase in d because the final solution is searched more intensively.

2. https://www.aliyun.com

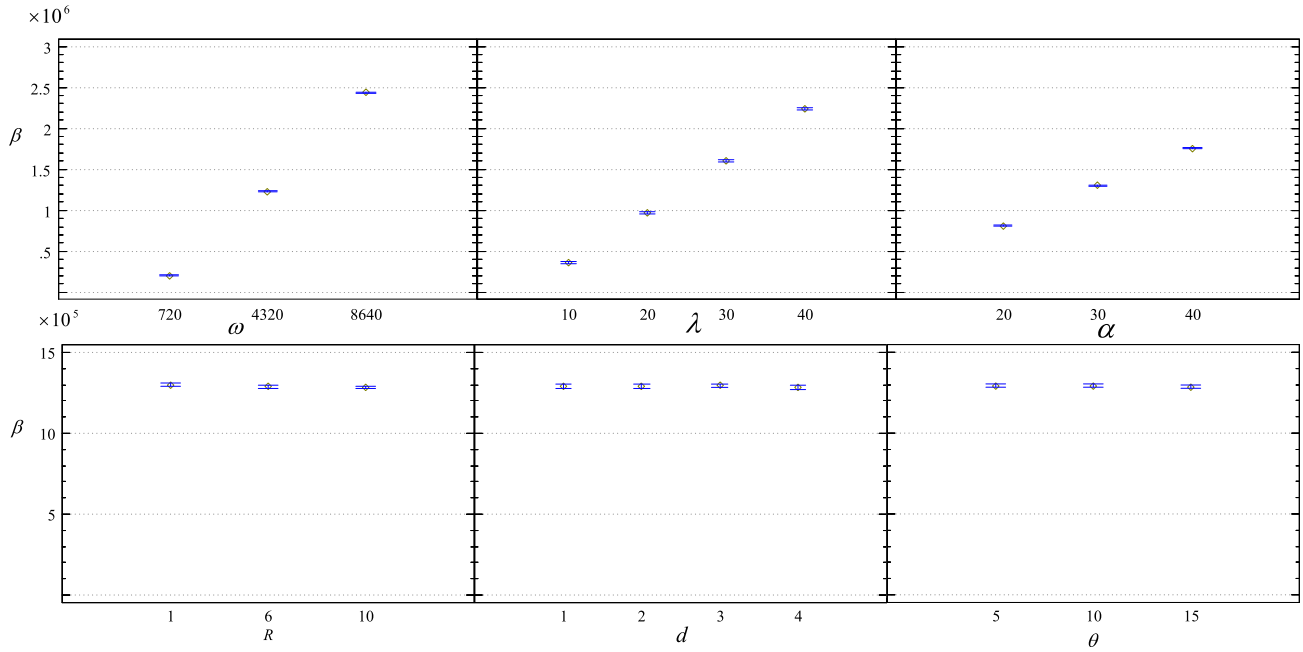


Fig. 3. The mean plots of the six studied parameters with 95 percent confidence level Tukey HSD intervals.

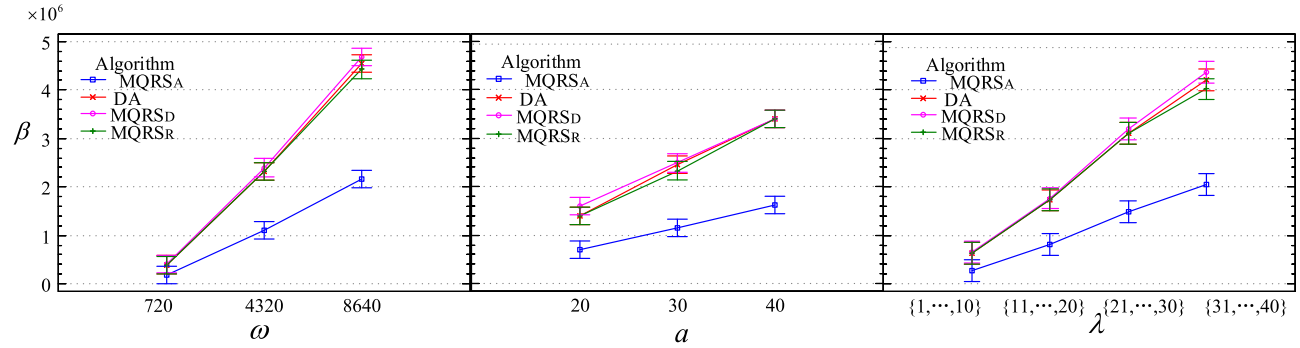


Fig. 4. The mean plots of the interaction between algorithms and parameters with 95 percent confidence level intervals.

(vi) The statistical differences of θ on β are insignificant. With an increase in θ, the impatience time decreases.

5.2 Algorithms Comparison

Implementing multiple parallel queues involves more design options compared to the single queues [34]. Unfortunately, few existing research considers stochastic requests allocated to a multi-queue system. A representative work is the distribution algorithm (DA) [21] that is proposed to solve the load distribution problem for multiple queues with a server in each queue. DA is adopted in one of non-observable parallel queues which does not consider the OSPI procedure. To evaluate the performance of the proposed algorithm, we compare four algorithms: MQRS_D, MQRS_A, MQRS_R and DA [21]. MQRS_D, MQRS_A and MQRS_R are based on the proposed MQRS framework with different request stream splitting processes. More specifically, MQRS_D splits the request stream using the RSS algorithm. MQRS_A averagely splits the request stream based on the number of server types while MQRS_R conducts the request stream according to the proportion which is the total service rate of all servers in each queue to that of the system. The four algorithms are compared using both randomly generated simulation instances and the real instances provided by AuverGrid[3] and the real production Cluster-trace-v2018[4] published by the Alibaba Group for parallel systems where requests are processed by reserved or on-demand servers.

5.2.1 Performance Comparison on Simulated Instances

Since there are no benchmark instances for the considered problem, we first compare the four algorithms across simulated instances. In terms of the calibrated results in Section 5.1, the system parameters ω (the time period for the reserved servers) $\in$\{720, 4,320, 8,640\}, α (income per request) $\in$\{20, 30, 40\}, λ (arrival rates of requests) $\in$\{\{1, ..., 10\}, \{11, ..., 20\}, \{21, ..., 30\}, \{31, ..., 40\}\}, $R = 6$ (queue capacity) and $\frac{1}{\theta} = 0.1$ (expected MWT of consumers) are tested along with the algorithm component $d = 2$. Five instances are randomly generated for each of the 36 combinations, i.e., 180 instances are tested on each of the four algorithms. The results are shown in Fig. 4.

From Fig. 4, we can observe that with an increase in ω, MQRS_D obtains the best profit. The profit of MQRS_R is

3. http://gwa.ewi.tudelft.nl/datasets/gwa-t-4-auvergrid
4. https://github.com/alibaba/clusterdata

TABLE 5
Comparison of the MQRS Algorithms and DA

	MQRS_A	DA[21]	MQRS_D	MQRS_R
β (Total profit)	1,155,410	2,418,690	2,496,470	2,376,320
CPU time (Second)	26	4787	227	203

smaller than that of MQRS_D while it is better than that of MQRS_R. MQRS_A always demonstrates the worst profit performance among MQRS_D, MQRS_R and DA. With an increase in α from 20 to 40, MQRS_D always obtains the largest profit whereas MQRS_A obtains the smallest. Only when α takes 40, there is small difference among MQRS_D, MQRS_R and DA algorithms. The results also indicate that a higher α value results in a better profit for MQRS_D.

When the arrival rate λ takes a value from $\{1, \ldots, 10\}$ and $\{11, \ldots, 20\}$, the profit obtained by MQRS_D is slightly larger than those of the other three algorithms. MQRS_R always obtains a higher profit than DA. In other words, with an increase in λ, MQRS_D is the most effective method. MQRS_A always shows the worst profit performance among the four compared algorithms.

To compare the algorithms comprehensively, the total profit and CPU times are shown in Table 5. From Table 5, it can be observed that the profits of MQRS_D, MQRS_R and MQRS_A are 2,496,470, 2,376,320 and 1,155,410, respectively. The profit of DA is 2,418,690 which is better than that of MQRS_R but worse than that of MQRS_D. MQRS_D uses 227s of computation time, which is acceptable in practice. The CPU time of MQRS_A and MQRS_R is 26s and 203s respectively because MQRS_A and MQRS_R are deterministic in the request splitting procedure. DA is the most time-consuming method which spends the longest CPU time even if it does not include the OSPI procedure.

5.2.2 Performance Comparison on Real Life Instances

To verify the performance of the proposed algorithm in real world applications, the VM workload data set provided by Microsoft Azure [43] and the real production Cluster-trace-v2018 published by the Alibaba Group are tested.

According to Microsoft Azure [43], the lifetime of 87.4 percent[5] of VMs is within 12.5 hours, i.e., the service rate of VMs is 0.08 per hour. To thoroughly evaluate the proposed MQRS, we take the service rate from {0.08, 0.17, 0.25, 0.33, 0.41, 0.5, 0.67, 0.75, 0.83, 0.92, 1, 3}. The arrival rate λ takes values from {1, 2, 3, ..., 40}. Based on μ_1, the prices for renting faster reserved and on-demand servers are $0.27(\frac{\mu_i}{\mu_1}+i)(i \in \{2, \ldots, S\})$ and $0.66(\frac{\mu_i}{\mu_1}+i)(i \in \{2, \ldots, S\})$ (¥) respectively. The other parameters are identical to those in the simulation. The comparison results on different algorithms are shown in Fig. 5. According to Fig. 5, $MQRS_A$ always obtains the minimum profit when $\lambda \leq 32$. When λ is between 1 and 18, the profit curves of $MQRS_D$, $MQRS_R$ and DA are intertwined, i.e., the superiority changes dynamically. When $\lambda > 32$, the profit of $MQRS_R$ fluctuates significantly because the proportion split results in local optimal solutions. In addition, when λ is bigger than 18, $MQRS_D$ outperforms the other two and $MQRS_R$ outperforms DA. These results are in accordance with those obtained in random instances.

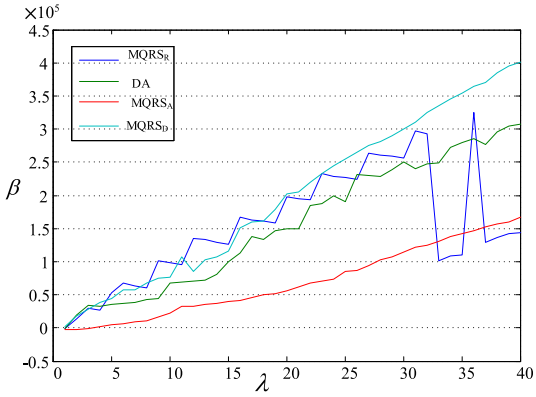


Fig. 5. Profit comparison among different algorithms.

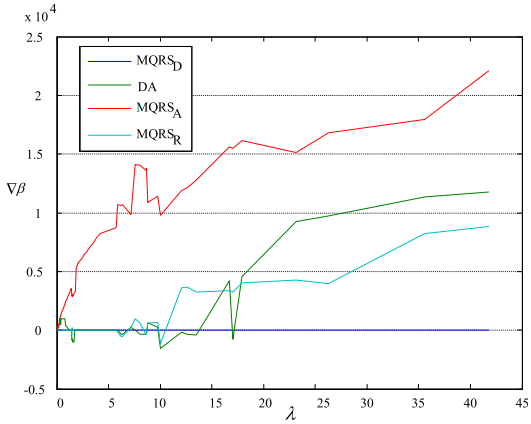


Fig. 6. Profit difference comparison among different algorithms and $MQRS_D$.

The real production Cluster-trace-v2018 published by the Alibaba Group, which contains eight-day sample data from one of the production clusters, is also tested. By analyzing the $start\_time$[6] of the requests, the arriving time intervals are obtained. The distributions of each request is Poisson and the execution time of each server is exponential with different arrival and service rates which have been calibrated in [31]. According to $start\_time$ and end_time,[7] the execution times of servers are calculated where service rates are 0.80, 1.69, 4.98, 6.77, 7.71, 10.54, respectively. Other parameters are identical to those in the simulation. The difference from the profit of $MQRS_D$ to all the algorithms are compared in Fig. 6. From the figure we can observe that with the increase of λ, the difference increases. When λ is between 1 and 18, the profit difference curves of $MQRS_D$, $MQRS_R$ and DA are intertwined, i.e., the superiority changes dynamically. When λ is bigger than 18, $MQRS_D$ outperforms the other two and $MQRS_R$ outperforms DA. These results are in accordance with those obtained in the Microsoft Azure instances.

5. https://azurecloudpublicdataset.blob.core.windows.net/azurepublicdataset/sosp_data/lifetime.txt

6. http://clusterdata2018pubcn.oss-cn-beijing.aliyuncs.com/batch task.tar.gz

7. http://clusterdata2018pubcn.oss-cn-beijing.aliyuncs.com/batch instance.tar.gz

6 Conclusion and Future Research

In this paper, a multi-queue system is constructed to maximize the total profit for service providers by specially considering impatient service consumers of e.g., call centers. An MQRS (Multi-Queue Request Scheduling) algorithm framework is proposed for the considered problem. Several methods are developed to determine the number of reserved servers and to calculate the probability of the expected number of on-demand servers. All parameters and components of the proposed algorithm framework are calibrated by the multi-factor analysis of variance statistical technique over a comprehensive set of random instances. The results show that for a long time period, the income per request and the arrival rates have a statistically significant influence, whereas the queue capacity and the delay rate have little impact, on the profit performance of the proposed algorithms. By comparing the proposed algorithms with the state-of-the-art algorithm DA (distribution algorithm) for a similar problem on both simulated and real life instances, our results demonstrate that: MQRS_A always obtains the worst profit among the four compared algorithms, MQRS_R is sometimes better than the others, and DA is generally worse than MQRS_D. MQRS_D mostly outperforms the other three algorithms across a comprehensive set of real life instances with acceptable computation times. The reason lies in that the proposed stream splitting method adaptively divides the request stream in terms of the service rates, i.e., all servers work in balanced loads.

For heterogeneous cloud centers, there are still many open research issues that are worth studying in the future, e.g., the arrival rates of requests could be relaxed to more practical distributions, and similarly for more realistic service rates of cloud servers.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB1400800, in part by the National Natural Science Foundation of Chinaunder Grants 61872077 and 61832004, and in part by the Collaborative Innovation Center of Wireless Communications Technology. The work of Quan Z. Sheng was supported in part by Australian Research Council Future Fellowship under Grant FT140101247 and in part by Discovery Project under Grant DP180102378. The work of Rubén Ruiz was supported in part by the Spanish Ministry of Science, Innovation, and Universities through the project OPTEP-Port Terminal Operations Optimization under Grant RTI2018-094940-B-I00 financed with FEDER funds.

References

[1] D. Gross, J. F. Shortie, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory*, 4th ed. Hoboken, NJ, USA: Wiley, 2013.

[2] J. Mei, K. Li, A. Ouyang, and K. Li, "A profit maximization scheme with guaranteed quality of service in cloud computing," *IEEE Trans. Comput.*, vol. 64, no. 11, pp. 3064–3078, Nov. 2015.

[3] Y. J. Chiang, Y. C. Ouyang, and C. H. Hsu, "Performance and cost-effectiveness analyses for cloud services based on rejected and impatient users," *IEEE Trans. Serv. Comput.*, vol. 9, no. 3, pp. 446–455, May/Jun. 2016.

[4] A. A. Bouchentouf and A. Guendouzi, "Cost optimization analysis for an $M^X/M/c$ vacation queueing system with waiting servers and impatient customers," *SeMA J.*, vol. 76, no. 2, pp. 309–341, 2019.

[5] E. Zohar, A. Mandelbaum, and N. Shimkin, "Adaptive behavior of impatient customers in tele-queues: Theory and empirical support," *Manag. Sci.*, vol. 48, no. 4, pp. 453–590, 2002.

[6] A. Mandelbaum, A. Sakov, and S. Zeltyn, "Empirical analysis of a call center," 2000. [Online]. Available: http://iew3. technion. ac. il/serveng/References/ccdata. pdf

[7] S. Zeltyn and A. Mandelbaum, "Call centers with impatient customers: Many-server asymptotics of the M/M/ n + G queue," *Queueing Syst. Theory Appl.*, vol. 51, no. 3–4, pp. 361–402, 2005.

[8] R. Ibrahim, "Managing queueing systems where capacity is random and customers are impatient," *Prod. Oper. Manag.*, vol. 27, no. 2, pp. 234–250, 2018.

[9] A. Movaghar, "Analysis of a dynamic assignment of impatient customers to parallel queues," *Queueing Syst.*, vol. 67, no. 3, pp. 251–273, 2011.

[10] M. Delasay, "Maximizing throughput in finite-source parallel queue systems," *Eur. J. Oper. Res.*, vol. 217, no. 3, pp. 554–559, 2012.

[11] M. Malawski, K. Figiela, and J. Nabrzyski, "Cost minimization for computational applications on hybrid cloud infrastructures," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1786–1794, 2013.

[12] V. G. Abhaya, Z. Tari, P. Zeephongsekul, and A. Y. Zomaya, "Performance analysis of EDF scheduling in a multi-priority preemptive M/G/1 queue," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 8, pp. 2149–2158, Aug. 2014.

[13] L. Aminizadeh and S. Yousefi, "Cost minimization scheduling for deadline constrained applications on vehicular cloud infrastructure," in *Proc. Int. Conf. Comput. Knowl. Eng.*, 2014, pp. 358–363.

[14] L. Chen, X. Li, and R. Ruiz, "Resource renting for periodical cloud workflow applications," *IEEE Trans. Serv. Comput.*, vol. 13, no. 1, pp. 130–143, Jan./Feb. 2020.

[15] H. Khazaei, J. Misic, and V. B. Misic, "Performance analysis of cloud computing centers using M/G/m/m+ r queuing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 5, pp. 936–943, May 2012.

[16] H. Khazaei, J. Misic, V. B. Misic, and S. Rashwand, "Analysis of a pool management scheme for cloud computing centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 5, pp. 849–861, May 2013.

[17] T. Atmaca, T. Begin, A. Brandwajn, and H. Castel-Taleb, "Performance evaluation of cloud computing centers with general arrivals and service," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 8, pp. 2341–2348, Aug. 2016.

[18] H. Khazaei, J. Misic, and V. B. Misic, "Performance of cloud centers with high degree of virtualization under batch task arrivals," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 12, pp. 2429–2438, Dec. 2013.

[19] C. Knessl, B. Matkowsky, Z. Schuss, and C. Tier, "Two parallel queues with dynamic routing," *IEEE Trans. Commun.*, vol. COMM-34, no. 12, pp. 1170–1175, Dec. 1986.

[20] B. Legros, "Waiting time based routing policies to parallel queues with percentiles objectives," *Oper. Res. Lett.*, vol. 46, no. 3, pp. 356–361, 2018.

[21] Y. Tian, C. Lin, and K. Li, "Managing performance and power consumption tradeoff for multiple heterogeneous servers in cloud computing," *Cluster Comput.*, vol. 17, no. 3, pp. 943–955, 2014.

[22] K. Li, "Optimal power allocation among multiple heterogeneous servers in a data center," *Sustain. Comput., Inform. Syst.*, vol. 2, pp. 13–22, 2012.

[23] H. Yuan, M. Zhou, Q. Liu, and A. Abusorrah, "Fine-grained resource provisioning and task scheduling for heterogeneous applications in distributed green clouds," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 5, pp. 1380–1393, Sep. 2020.

[24] H. Yuan, J. Bi, and M. Zhou, "Geography-aware task scheduling for profit maximization in distributed green data centers," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2020.3001051.

[25] K. Li, C. Liu, K. Li, and A. Y. Zomaya, "A framework of price bidding configurations for resource usage in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 8, pp. 2168–2181, Aug. 2016.

[26] J. Mei, K. Li, and K. Li, "Customer-satisfaction-aware optimal multiserver configuration for profit maximization in cloud computing," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 1, pp. 17–29, Jan./Mar. 2017.

[27] J. Cao, H. Kai, K. Li, and A. Y. Zomaya, "Optimal multiserver configuration for profit maximization in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1087–1096, Jun. 2013.

[28] H. Yuan, J. Bi, W. Tan, and B. H. Li, " Temporal task scheduling with constrained service delay for profit maximization in hybrid clouds," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 1, pp. 337–348, Jan. 2017.

[29] P. Zhang and M. Zhou, " Dynamic cloud task scheduling based on a two-stage strategy," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 772–783, Apr. 2018.

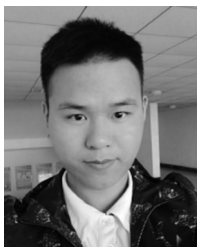
[30] S. Hosseinalipour and H. Dai, "Options-based sequential auctions for dynamic cloud resource allocation," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.
[31] S. Wang, X. Li, and R. Ruiz, "Performance analysis for heterogeneous cloud servers using queueing theory" *IEEE Trans. Comput.*, vol. 69, no. 4, pp. 563–576, Apr. 2020.
[32] H. Khazaei, J. Misic, and V. B. Misic, " Performance of cloud centers with high degree of virtualization under batch task arrivals," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 12, pp. 2429–2438, Dec. 2013.
[33] R. R. Inman, "Empirical evaluation of exponential and independence assumptions in queueing models of manufacturing systems," *Prod. Oper. Manag.*, vol. 4, no. 8, 409–432, 1999.
[34] A. Psarras, M. Paschou, C. Nicopoulos, and G. Dimitrakopoulos, "A dual-clock multiple-queue shared buffer," *IEEE Trans. Comput.*, vol. 66, no. 10, pp. 1809–1815, Oct. 2017.
[35] A. Movaghar, "On queueing with customer impatience until the beginning of service," *Queueing Syst.*, vol. 29, pp. 337–350, 1998.
[36] H. C. Tijms, *A First Course in Stochastic Models*. Amsterdam, The Netherlands: Wiley, 2003.
[37] B. W. Wah and T. Wang, "Efficient and adaptive lagrange-multiplier methods for nonlinear continuous global optimization," *J. Glob. Optim.*, vol. 14, pp. 1–25, 1999.
[38] D. G. Luenberger, *Optimization by Vector Space Methods*. Stanford, CA, USA: Wiley, 1969.
[39] J. Nino-Mora, "Overcoming numerical instability in one-step policy improvement for admission and routing to queues with firm deadlines," in *Proc. Int. Conf. Netw. Games, Control Optim.*, 2014, pp. 127–134.
[40] K. R. Krishnan, "Joining the right queue: A Markov decision-rule," in *Proc. IEEE Conf. Decis. Control*, 1987, pp. 1863–1868
[41] J. L. Diarialgo, M. Garcia, J. Garcia, and D. F. Garcia, " Optimal allocation of virtual machines in multi-cloud environments with reserved and on-demand pricing," *Future Gener. Comput. Syst.*, vol. 71, pp. 129–144, 2017.
[42] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, pp. 268–308, 2001.
[43] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proc. 26th Symp. Operating. Syst. Principles*, 2017, pp. 153–167.

Shuang Wang received the BSc degree from the College of Sciences,Nanjing Agricultural University, in 2015 and the PhD degree from the School of Computer Science and Engineering, Southeast University, Nanjing, China, in 2020. From 2019 to 2020, she was a visiting PhD student with the Department of Computing, Macquarie University, Sydney, Australia. She is currently a postdoctoral research fellow with the Department of Computing, Macquarie University. She has authored or coauthored several papers on international journals and conferences including the *IEEE Transactions on Computers* and International Conference on Service Oriented Computing. Her research interests include cloud computing, task scheduling, performance analysis, and truth discovery.

Xiaoping Li (Senior Member, IEEE) received the BSc, MSc, and PhD degrees in applied computer science from the Harbin University of Science and Technology, Harbin, China, in 1993, 1999, and 2002, respectively. He is currently a distinguished professor with the School of Computer Science and Engineering, Southeast University, Nanjing, China. He has authored or coauthored more than 100 academic papers, some of which have been published in international journals including the *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Cybernetics*, *IEEE Transactions on Automation Science and Engineering*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Systems, Man and Cybernetics: Systems*, *Information Sciences*, *Omega*, and the *European Journal of Operational Research*. His research interests include scheduling in cloud computing, scheduling in cloud manufacturing, service computing, big data, and machine learning.

Quan Z. Sheng received the PhD degree in computer science from the University of New South Wales. He was a postdoc research scientist with CSIRO ICT Centre. He is currently a full professor and the head of the Department of Computing, Macquarie University, Sydney, Australia. He has authored or coauthored more than 400 publications. His research interests include service oriented computing, distributed computing, data analytics, internet computing, and Internet of Things. He has ranked by Microsoft Academic as one of the most impactful authors in Services Computing. He was the recipient of the AMiner Most Influential Scholar Award in IoT in 2019, the ARC Future Fellowship in 2014, the Chris Wallace Award for Outstanding Research Contribution in 2012, and the Microsoft Fellowship in 2003.

Rubén Ruiz is currently a full professor of statistics and operations research with the Universitat Politècnica de València, Spain. He has coauthored more than 80 papers in International Journals and has participated in presentations of more than a hundred papers in national and international conferences. He is the editor of the Elsevier's journal *Operations Research Perspectives* and a coeditor of the JCR-listed journal *European Journal of Industrial Engineering*. He is also an associate editor for other important journals including *TOP* and also a member of the editorial boards of several journals most notably *European Journal of Operational Research* and *Computers and Operations Research*. He is currently the director of the Applied Optimization Systems Group with the Universitat Politècnica de València, where he has been a principal investigator of several public research projects and privately funded projects with industrial companies. His research interests include scheduling and routing in real life scenarios.



Jinquan Zhang received the BSc degree from the College of Science, Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2017. He is currently working toward the PhD degree with the School of Computer Science and Engineering, Southeast University, Nanjing, China. His research interests include reinforcement learning, combinatorial optimization, and cloud resource scheduling.

Amin Beheshti received the master's and bachelor's degrees in computer science with First Class Honours and the PhD and Postdoc degrees from the School of Computer Science and Engineering, UNSW Sydney. He is currently the director of AI-enabled Processes Research Center and the head of the Data Analytics Research Lab, Department of Computing, Macquarie University. He is currently a senior lecturer of data science with Macquarie University and an adjunct academic in computer science with UNSW Sydney. He is the leading author of the book titled *Process Analytics*, and has coauthored with other high-profile researchers in UNSW and IBM research, published by Springer. He is or was invited as a keynote speaker, general-chair, PC-Chair, organization-chair, and program committee member of top international conferences.