ABSTRACT The ternary optical computer (TOC) has attracted increasing attention from its providers and potential customers because of the advantages of its optical processor, such as low power consumption, numerous trits, parallelism, dynamical reconfigurability and bitwise allocability. The analysis of its performance has become an urgent problem to be solved in recent years. This paper builds a four-stage TOC service model by introducing synchronous multi-vacations and tandem queueing. Here, a vacation refers specifically to an optical processor vacation. Additionally, we propose a processor-divided-equally (PDE) strategy and a task scheduling and optical processor allocation algorithm under this strategy. Assuming that the intervals of task arrival follow a homogeneous Poisson process, we obtain some important performance indicators, such as the mean response time, mean number of tasks, and optical processor utilization under the PDE strategy, which are based on the M/M/1 and M/M/n queueing systems with exhaustive service and synchronous multi-vacations. The numerical results illustrate that the number of small optical processors has an important effect on the performance of TOC and that high vacation rate can improve the system performance to some extent.

INDEX TERMS Synchronous multi-vacations, tandem queueing, exhaustive service, response time, processor-divided-equally strategy.

I. INTRODUCTION
In recent decades, to overcome the bottlenecks of electronic computers such as computing speed, bandwidth and power, researchers have increasingly focused on optical computing. For instance, Zangeneh-Nejad et al designed a reconfigurable and highly miniaturized differentiator to perform analog optical differentiation by using a half-wavelength plasmonic graphene film [1]. Babashah et al. proposed a fully reconfigurable on-chip photonic signal processor with a bandwidth of 400 GHz based on dispersive Fourier transform to perform differentiation, integration and convolution in the time domain [2]. Yue et al. proposed an optical computation framework to remove non-uniform blur caused by camera shaking by using an off-the-shelf projector and a camera mounted on a programmable motion platform [3]. Pang et al. presented a layered optical logic element which used wavelength division multiplexing for its inputs and output, and implemented optical nanoscale computing [4]. In addition, Jin et al. proposed the principle and architecture of a ternary optical computer (TOC), which expresses information in three optical states (horizontally polarized light, vertically polarized light, no intensity light) [5], [6]. These optical computing platforms all took full advantage of the multidimensional parallel nature of light.

A lot of significant benefits have been obtained since the proposal of TOC. For example, Yan et al. proposed the decrease-radix design principle [7], which not only makes the construction of the TOC processor normative but also makes the processor fully reconfigurable and bitwise allocable in runtime [8]–[10]. Based on the principle, Jin et al. built an

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
up-to-date TOC with 1152 ternary data-bits, i.e., trits by combining six identical TOCs in 2018, called SD2018. Wang et al. implemented a fully parallel carry-free addition in three steps based on the modified-signed-digit(MSD) arithmetic on a TOC and performed vector-matrix multiplication by using the binary-addition tree algorithm [8]. These addition and multiplication methods inaugurated the application of TOC in numerical calculations. In addition, Jin and Peng et al. presented the relevant theory of a TOC adder based on MSD arithmetic [9]–[11]. A dual-space storage technique was proposed for improving the speed of accessing data in a TOC optical processor [12]. Xu et al. designed and implemented a multiplication routine based on the TOC carry-free adder [13]. Zhang et al. proposed a data computing model and formulated a programming method on the TOC [14], which was convenient for customers to use. In addition, a double-rotator-structure for the TOC optical processor and its control mechanism were presented for decreasing the I/O communication overhead [15]. It can be seen that breakthroughs have been made in software and hardware as well as numerical applications.

However, there are few reports on another important research direction for TOC—performance analysis and evaluation other than our works [16]–[19]. In our works, a four-stage service model for TOC performance analysis and evaluation is built, and response time is chosen as a performance metric. However, they are based on different queueing systems. Reference [16] built an analytic model for analyzing and evaluating the performance of TOC based on a tandem M/M/1 queueing system. Its results show that the speed of the network is the bottleneck of response time. Under instant-scheduling and end-scheduling strategies, [17] and [18] put M/M/1, M/M/n, M^X/M/1 and M/M^B/1 together in a series, building a complex tandem queueing system, for obtaining the response time. The conclusion is that the end-scheduling strategy is superior to the instant-scheduling strategy. However, these models cannot accurately reflect the computing ecology of TOC. For instance, they assumed that TOC will not fail. In other words, TOC does not need repair, apparently, TOC is in an ideal state. As a consequence, [19] built a service model taking synchronous TOC vacations into consideration to obtain the response time. The results illustrated that the number of small optical processors and the vacation rate have an important and minor effect on the response time, respectively. However, these works only obtained the response time and did not delve further into other performance indicators, such as the number of tasks and utilization of the optical processor. Therefore, this paper will propose a processor-divided-equally(PDE) strategy, design a task scheduling and processor allocating algorithm under this strategy, and build a performance analysis model by connecting several M/M/1 queueing systems and an M/M/n queueing system on the basis of previous works.

In this paper, we build a complex queueing model to investigate the performance of a TOC, considering its vacations. We aim to make the following contributions in this paper:

- In this paper, we first demonstrate the client/server computing paradigm of TOC. The server consists of receiving module(RM), preprocessing module(PPM), scheduling and allocating module (SAM), optical processor, decoder, and transmitting module(TM).
- We illuminate a method of building a TOC service model based on queue with vacations and tandem queue. The whole optical processor of a TOC will be on vacations when there is no request to be computed and all requests or tasks are accomplished, i.e., exhaustive service. In addition, the vacation will not be over until there is at least one request to be scheduled and there has been at least one vacation. Otherwise, the TOC will go on a vacation. We model the TOC as a complex queueing system to illustrate the service process. Four concatenated queueing systems compose the complex queueing model. The first is the infinite capacity receiving queueing system, which is used to receive operation requests. The second is the preprocessing queueing system, which is used to generate the control signal of the optical processor. The third is the scheduling queueing system, which is used to schedule tasks, allocate resources, and execute computations on the TOC. The fourth is the transmitting queueing system, which is used to send the computing results to the corresponding customer.
- Inspired by SD2018, i.e., the easy extensibility of TOC, we propose a PDE strategy. In other words, the whole optical processor can be divided into many small identical optical processors for customer use. Additionally, we design a task scheduling and processor allocation algorithm with vacations under this strategy.
- Based on the complex queueing model, we analyze the real output of the TOC, that is, the mean response time of the system, the mean number of tasks in the system, and other performance indicators, by use of queueing theory. We also use these indicators to evaluate the performance of the TOC.
- We conduct extensive simulations to evaluate and analyze the performance of the TOC. Moreover, the numerical results demonstrate that the proposed service model is theoretically correct and that the models of performance indicators are self-consistent and faithfully reproduce the computing ecology of TOC from which they were built.

The rest of the paper is organized as follows. In Section 2, we chiefly describe some queueing systems, in particular the queueing system with vacations. In Section 3, we introduce the computing paradigm of TOC and the service model based on queueing system with vacations. In Section 4, we mainly present a task scheduling and processor allocation algorithm under the PDE strategy. Section 5 focuses on the construction of the performance analysis and evaluation model. Section 6 demonstrates the simulation results and
analysis of the choice indicators. Finally, Section 7 gives some concluding remarks and possible future research.

II. BACKGROUND AND MOTIVATION
The queueing system is an effective tool for analyzing and evaluating system performance. The tandem queueing system, an extraordinary queueing network, is constructed by connecting several queueing systems sequentially. And it is also widely used for performance analysis and evaluation of manufacturing lines [20], computer systems [21], [22] and communication networks [23]–[26].

A queueing system with vacations is a system in which service facilities are repaired or attendants go on vacations during idle time. As shown in Fig. 1, the oblique arrows above and below the horizontal line indicate the task arrivals and the departures after being serviced, respectively. A complete vacation policy consists of not only the rules of the start and end of the vacation but also the distribution of vacation time. They can be divided into empty exhaustion service policies [27], [28] and non-empty exhaustion service policies [29] according to the rule of governing the start of vacations. In an empty exhaustion service system, only when there is no task can it start a vacation, while in a non-empty exhaustion service, the system can take vacations if there are tasks to be serviced. At the same time, policies can be divided into single vacation and multiple vacation policies according to the rule of vacation end [27]–[32]. With a multiple vacation policy, the system will not finish a vacation and begin service until one vacation is over and there are tasks to be done. Otherwise, the next vacation among the independent and identically distributed vacations will be restarted [31]–[33]. Therefore, compared with the classical queueing system, the queueing system with vacations can more truthfully reflect the fact that service is interrupted and provide flexibility for the process control and the optimization design of the system.

In recent decades, many researchers have broadly investigated queueing systems with vacations and proposed some new approaches, such as matrix analysis [34], [35] and matrix geometry [33], [36]. Meanwhile, the queueing system with vacations has been applied to manufacturing systems [30], [37], polling systems [38] and cloud computing [39], [40] for performance analysis and evaluation. These models of performance analysis and evaluation, especially for cloud computing, cannot be applied to TOC directly because of the characteristics of TOC optical processor, such as having many trits and being dynamically bitwise reconfigurable and bitwise allocatable. Therefore, this paper will build a service model of TOC based on the tandem queueing system with vacations, present a task scheduling and processor allocation algorithm under the PDE policy, select essential indicators such as the mean response time and the system mean task number, and analyze the performance of TOC by using the M/M/1 queueing system and M/M/n queueing system with synchronous multi-vacations.

III. SERVICE MODEL OF TOC BASED ON A TANDEM QUEUEING SYSTEM WITH VACATIONS

A. COMPUTING PARADIGM OF TOC
The computing paradigm of TOC is client/server, shown in Fig. 2. The system is made up of a client and server, and the server consists of a master computer and a slave computer. The former is composed of a receiving module(RM), preprocessing module(PPM), scheduling and allocating module(SAM), transmitting module(TM), and the latter is composed of an optical processor(OP) and decoder(DC). Moreover, the tasks carried out in the TOC will perform the following steps.

- A customer submits an operation request, i.e., task, via the client. The client sends it to the RM after calculating the number of binary tri-valued logic operations needed, the operation amount of each logical operation, the total operation amount after the customer finishes inputting the operations and operands.
- The RM sends the received request to the PPM.
- The PPM preprocesses the data, i.e., operands. In other words, it transforms the data into the control signals of the optical processor. Then, it sends the request to the SAM.
- The SAM is responsible for scheduling tasks, allocating the trits of the optical processor, looking up the reconfiguration codes of the needed operators, and sending them to the TOC, i.e., the slave computer.
- The optical processor performs computations after the operators are reconfigured in parallel. The decoder handles decoding, that is, transforming the optical signals into electrical signals. The results are sent to the optical processor or TM [8]–[11].
- Finally, the TM transforms the electrical signals into the final results understood by customers, and sends the results to the corresponding client.

B. SERVICE MODEL FOR TOC PERFORMANCE ANALYSIS
To analyze and evaluate the performance of a TOC, we first build its service model based on the queueing system. The model is shown in Fig. 3. The model is made up of four concatenated queueing systems. They are the receiving queueing system, the preprocessing queueing system, the scheduling queueing system, and the transmitting queueing system.
queueing system and the transmitting queueing system in turn. In addition, they constitute a complex queueing model that manifests the homogeneity of the service processes and the optical processor of the TOC. We will obtain the performance metrics including the mean response time $T$, the mean number of requests $R$ in the system, the utilization $U$ of the optical processor, and the vacation probability $P$ of the TOC. Assume that the four queues are all blocked-request delay and the queue policy is first-come-first-served (FCFS). The model is depicted below.

In the first queueing system, i.e., the receiving queueing system, the RM receives a request submitted by a customer from the first-level queue, i.e., the request queue according to the FCFS policy when the queue is not empty and change it into a pending task on the TOC by sending it to the PPM.

In the second queueing system, the PPM similarly inserts the tasks sent by the RM into the second-level queue where they wait to be preprocessed, and gets a task from the non-empty queue according to the FCFS policy. Then, it transforms the decimal data input by the user into MSD data since the operands which can be directly processed are MSD operands and finds the needed binary tri-valued logic operations for achieving the request or task. Finally, it sends the task to the SAM after generating the control signals of the OP according to the MSD data and the logic operations, and gets another task from the non-empty preprocessing queue.

The third queueing system is very different from the other queueing systems. The other queueing systems are made up of only one module each, as shown in Fig. 3. However, the scheduling queueing system driven by the SAM consists of several components such as the SAM and slave computer, as shown in Fig. 3. We shall explore the TOC service model based on exhaustive service and multi-vacations in detail. In this model, the optical processor is divided equally into several small OPs and they have three states: leisure, busy, on vacation. Meanwhile, their vacations are synchronous. The complete workflow for this stage is shown in Fig. 4.

In the final queueing system, i.e., the transmitting queueing system, the TM puts the computing results in MSD into the final level queue, i.e., the result queue. It also obtains the customer’s computing results in turn from the non-empty queue, and sends them to the corresponding client after transforming them into a form that can be understood by the customer. Moreover, a vacation takes place when there is no task to be processed in the slave computer and no task to be scheduled in the SAM. At the moment, all the small OPs are on vacation. Meanwhile, the vacation that is coming to an end no free small OP. In addition, the slave computer performs parallel computing, i.e., calculates multiple data simultaneously, by use of the control signals generated by the PPM after the reconfiguration unit (RU) reconfigures the operator in parallel according to the allocating information and the reconfiguration codes. The DC decodes the computing results represented in optical signals, i.e., changes them into MSD data. And the DC sends the decoding results to the OP or TM after judging whether they take part in the next logic operation. Meanwhile, the slave computer checks whether there are data to be calculated. If so, it continues acquiring data; otherwise, it sends an “A task has been completed” signal to the SAM. The SAM checks if there are tasks to be scheduled after receiving the signal. If so, the SAM schedules a task and allocates resources for it once more; otherwise, the SAM sends a “Vacation” signal to the TOC. The TOC starts a vacation with a stochastic span after receiving the signal. And the TOC sends a “The vacation is over” signal to the SAM when the vacation is over. The SAM checks whether there are tasks to be scheduled after receiving the signal and takes the appropriate action.

In the final queueing system, i.e., the transmitting queueing system, the TM puts the computing results in MSD into the final level queue, i.e., the result queue. It also obtains the customer’s computing results in turn from the non-empty queue, and sends them to the corresponding client after transforming them into a form that can be understood by the customer. Therefore, the vacation in Fig. 3 is not the server vacation but the slave computer vacation. We call it the TOC vacation in brief.

Moreover, a vacation takes place when there is no task to be processed in the slave computer and no task to be scheduled in the SAM. At the moment, all the small OPs are on vacation. Meanwhile, the vacation that is coming to an end
will likely be followed by the next vacation after a random amount of time. In other words, the vacations are synchronous multi-vacations when the slave computer has finished all tasks.

IV. TASK SCHEDULING AND OPTICAL PROCESSOR ALLOCATING ALGORITHM WITH VACATIONS

The task scheduling strategy and resource allocation strategy in a TOC are different from those of other parallel computing platforms, such as cloud computing, because of the unique advantages of a TOC. This paper proposes a PDE strategy, and a task scheduling and processor allocating algorithm with vacations under this strategy. The task scheduling process consists of not only dispatching the first task in scheduling and a task scheduling and processor allocating algorithm with vacations. This paper proposes a PDE strategy, platforms, such as cloud computing, because of the unique performance of the TOC when it is in equilibrium. The equilibrium task number \( R \) and the equilibrium response time \( T \) in

Algorithm 1 Task scheduling and processor allocating algorithm under the PDE strategy

- Step 1: Initialize the system parameters. Let the number \( N_{\text{Proc}} \) of tasks being processed and the states \( \text{State}[i] (i = 0, 1, \ldots, n-1) \) of all small optical processors be zero. And \( L_Q = 0, i = 0 \) (i is used to point to the small OP to be allocated currently).
- Step 2: Jump to Step 3 when a task arrives.
- Step 3: Put the new arrival task into \( Q \), increase \( L_Q \) by 1, and judge whether \( L_Q \) is non zero. If so, judge whether the TOC is on vacation. If so, jump to Step 14; otherwise, judge whether \( N_{\text{Proc}} \) is equal to \( n \). If so, jump to Step 11; otherwise, jump to Step 4.
- Step 4: Schedule a task, increase \( N_{\text{Proc}} \) by 1, and jump to Step 5.
- Step 5: \( i = i \mod n \), check the state of the \( i \)th small OP; i.e., judge whether \( \text{State}[i] \) is equal to 0. If so, set \( \text{State}[i] \) to 1, and jump to Step 7; otherwise, jump to Step 6.
- Step 6: Increase \( i \) by 1, and jump to Step 5.
- Step 7: \( j = 1 \).
- Step 8: Judge whether \( j \) is greater than \( N_{\text{Log}} \). If so, jump to Step 10; otherwise, jump to Step 9.
- Step 9: Allocate the trits in proportion. That is, \( N_j = \lfloor N_{\text{DB}} \cdot C_j / C \rfloor \) \((j = 1, 2, \ldots, N_{\text{Log}})\). Increase \( j \) by 1 and jump to Step 8.
- Step 10: Send the allocating results, i.e., \( N_j (j = 1, 2, \ldots, N_{\text{Log}}) \) and the reconfiguration codes of processors to the slave computer, and jump to Step 14.
- Step 11: Decrease \( N_{\text{Proc}} \) by 1 when the SAM receives the signal, “A task has been completed”. In addition, reclaim the idle small OP, that is, \( \text{State}[i] = 0 \), and jump to Step 12.
- Step 12: Judge whether \( L_Q \) is equal to zero. If so, send the signal “Vacation” and jump to Step 14; otherwise, jump to Step 4.
- Step 13: Jump to Step 12 when the SAM receives the signal “The vacation is over”.
- Step 14: The algorithm ends.

V. ANALYTIC MODEL FOR PERFORMANCE ANALYSIS AND EVALUATION

In this section, we select principal indicators, such as the mean response time, mean task number, utilization of the optical processor and vacation probability, for evaluating the performance of the TOC when it is in equilibrium. The equilibrium task number \( R \) and the equilibrium response time \( T \) in

the TOC are the sum of the task numbers \( R_i (i = 1, 2, \ldots, 4) \) and the response times \( T_i \) of the four stages, respectively. That is, their equations can be formulated as

\[
R = \sum_{i=1}^{4} R_i, \quad T = \sum_{i=1}^{4} T_i
\]  

(1)

This queuing model can simplify system analysis and make it easier to deduce an equation for the important performance metrics (e.g., mean of tasks in the system, mean response time). Therefore, we will use the complex queuing model with the four concatenated queuing systems proposed...
in Section III.B to analyze the performance problems of a TOC.

A. RECEIVING QUEUEING SYSTEM

The RM in the first queueing system is used to receive the requests sent by users. To obtain $R_1$ and $T_1$, we can treat the first stage as an M/M/1 queueing system. In other words, the stream of requests is a Poisson process with arrival rate $\lambda$; that is, the inter-arrival times are independent and exponentially distributed with the parameter $1/\lambda$, and the receiving times are independent and exponentially distributed with the parameter $1/\mu$. Thus, the state-transition diagram for continuous-time Markov chain (CTMC) of the first queueing model is shown in Fig. 5, where $m$ is the request number in the queueing system. A request is being received and the other $m-1$ requests are waiting to be received.

The receiving queueing system is in equilibrium when $\rho = \lambda/\mu_1 < 1$ [41], [42]. Denote the steady-state probability in state $m$ under the equilibrium of the system as $p^*_m (m=0, 1, 2, \ldots)$. Based on Fig. 5, the steady-state balance equations of the system can be obtained as follows.

$$
\begin{align*}
p^*_0 &= \mu_1 p^*_1, \\
(\lambda + \mu_1) p^*_m &= \lambda p^*_{m-1} + \mu_1 p^*_{m+1}, \quad m \geq 1.
\end{align*}
$$

Then

$$p^*_m = \rho^m_1 p^*_0, \quad m \geq 1.$$  

We can obtain the idle probability $p^*_0$ of the RM using the normalization equation $\sum_{m=0}^\infty p^*_m = 1$.

$$p^*_0 = 1 - \rho_1.$$  

Then, the mean number of requests in the receiving queueing system is obtained.

$$R_1 = \sum_{i=0}^\infty i p^*_i = \rho_1 (1-\rho_1) \sum_{i=0}^\infty i p^*_{i-1}$$

$$= \rho_1 (1-\rho_1) \left( \frac{\rho_1}{1-\rho_1} \right) = \frac{\rho_1}{1-\rho_1} = \frac{\lambda}{\mu_1 - \lambda}. \quad (2)$$

Applying Little’s law (the response time $t = \text{response time of the} \ \text{receiving queueing system}$), we obtain the mean response time of the receiving queueing system

$$T_1 = \frac{R_1}{\lambda} = \frac{1}{\mu_1 - \lambda}, \quad (3)$$

where $\lambda$ and $\mu_1$ are the mean arrival rate of requests and the mean receiving rate of the RM per unit time, respectively. Denote the average amount of data for transmission in requests as $D$ and the average transmission speed of the network as $\xi$. Thus, $\mu_1 = \xi / D$. Substitute it into (2) and (3) to obtain

$$R_1 = \frac{\lambda}{D - \lambda}, \quad T_1 = \frac{1}{D - \lambda}. \quad (4)$$

B. PREPROCESSING QUEUEING SYSTEM

According to Burke’s theorem [43]–[45], the output of the first queueing system is also a Poisson process with the parameter $\lambda$ when it reaches equilibrium. That is, the mean arrival rate of requests in the second queueing system is equal to that in the first queueing system, i.e., $\lambda$. Obviously, the second queueing system can be represented similarly to an M/M/1 queueing system. The state-transition diagram for the second queueing system is similar to Fig. 5. Denote the preprocessing speed of the PPM as $\tau$; then the service rate $\mu_2$ is equal to $\tau / D$. The stationary probability equations for the preprocessing queueing system are identical to those for the receiving queueing system when $\rho_2 = \lambda / \mu_2 = \lambda D / \tau < 1$. Therefore, the mean request number $R_2$ and the mean response time $T_2$ in the preprocessing queueing system can be obtained by applying formula (4).

$$R_2 = \frac{\lambda}{D - \lambda}, \quad T_2 = \frac{1}{D - \lambda}. \quad (5)$$

C. SCHEDULING QUEUEING SYSTEM

The OP of the TOC has a large number of trits. For instance, a TOC with 1152 trits was built in 2018. Moreover, it was made up of six small OPs, and each of them had 192 trits. For this reason, under the PDE strategy the trits of the TOC are divided equally into $n$ parts, and each part is a small OP which can be used exclusively. Clearly, these small OPs are homogeneous. That is, they have the same hardware configuration—reconfigurable OPs with identical trits—and have the same computing power. Therefore, the third queueing system can be modeled as an M/M/n queueing system. Additionally, we consider synchronous multi-vacations and exhaustive service for performance analysis and evaluation of the TOC. Evidently, the arrival rate of tasks is also $\lambda$. In addition, we assume that the busy period of the TOC, the inter arrival time in the scheduling queueing system and the vacation time of the TOC are independent of each other.

As mentioned above, if and only if $Q$ is empty and all the small OPs are idle will these $n$ small OPs simultaneously start a vacation of stochastic length $\tau$. The tasks arriving during vacations are inserted into the tail of $Q$ in order. The slave computer sends the signal “The vacation is over” to the SAM when a vacation is over. The TOC will restart the next independent and identically distributed vacation if it receives the signal “Vacation”. Otherwise, the TOC will be busy after the vacation is over. If there are $L_Q < n$ tasks in $Q$ at this moment, the SAM schedules all the tasks according to Algorithm 1, the $L_Q$ small OPs begin optical computing and the other $n-L_Q$ OPs remain idle. If $L_Q \geq n$, $n$ tasks are scheduled and the rest of the tasks continue waiting in $Q$.  

FIGURE 5. State transition probability diagram for the receiving queueing model based on the M/M/1 queueing system.
According to [8]–[10], the computation amount $C_{\text{amount}}$ of each task is far greater than the data amount $D$. To simplify performance analysis, let $C_{\text{amount}} = 20D$. Although every trit of the OP needs reconfiguration, the reconfiguration time can be ignored, because the reconfiguration is parallel and takes a very short time. Assume that the vacation time $v$ follows an exponential distribution with the parameter $1/\delta$ (where $\delta$ is the vacation rate) and denote the computing speed of the TOC as $\sigma$. Then, the service rate of each small OP $v = \sigma/20D$. Thus, the service rate of each small OP $v = \sigma/20D$.

Now, we solve $R_3$ and $T_3$ in equilibrium based on the quasi-birth and death process model [33], [36]. Denote the task number in the third queueing system at time $t$ as $Q(t)$ and denote the states of the slave computer at time $t$ as $V(t)$, i.e.,

$$V(t) = \begin{cases} 0, & \text{if TOC is busy at time } t, \\ 1, & \text{if TOC is on vacation at time } t. \end{cases}$$

Thus, $\{(Q(t), V(t))\}$ forms a two-dimensional Markov process with the following state space:

$$\Omega = \{(0, 1) \cup \{(k, j) \mid k \geq 1, k \in \mathbb{N}, j = 0, 1\}.$$  

The states will change when a new task reaches the SAM, the TOC finishes a task or the vacation is over. We can obtain the state transition diagram shown in Fig. 6 by sorting the states according to the task number in the scheduling queueing system and the state for whether the TOC is on vacation. For example, state (2,1) means that the TOC is on vacation and there are two tasks in $Q$. It will transition to (3,1) with the rate $\lambda$ if a new task arrives during the vacation; it will transition to (2,0) with the rate $\delta$ if the vacation is over before the next task arrives. The state (2,0) represents that the TOC is busy, two tasks occupy two small OPs that run with service rate $2\mu_{3E}$, and the other $n-2$ small OPs are idle. In other words, it transitions to (1,0) with the rate $2\mu_{3E}$. Especially, for the state (1,0), the TOC will be on vacation if no task arrives after a small OP completes its task. That is, (1,0) transitions to (0,1) with the rate $\mu_{3E}$.

We can obtain the following generator matrix $G$ of Fig. 6 by writing the states of the process $(Q(t), V(t))$ as composite vectors in lexicographical order $G$, as shown at the bottom of this page, where $\omega_i^1 = -(i\mu_{3E} + \lambda)i \geq 1$, $\omega_i^* = -(\delta + \lambda)$. And $G$ can be written as the Block-Jacobi matrix

$$G = \begin{pmatrix} A_0 & C_0 \\ B_1 & A_1 & C_1 \\ B_2 & A_2 & C_2 \\ B_3 & A_3 & C_3 \\ \vdots & \vdots & \vdots \end{pmatrix}.$$ 

$$A_i = \begin{bmatrix} -(\lambda + i\mu_{3E}) & \delta & 0 \\ 0 & -\lambda - \delta & 0 \\ 0 & 0 & -\lambda \end{bmatrix}, 1 \leq i \leq n,$$

$$B_i = \begin{bmatrix} i\mu_{3E} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, 2 \leq i \leq n,$$

$$A = \begin{bmatrix} -(\lambda + n\mu_{3E}) & 0 & 0 \\ 0 & -(\lambda + \delta) & 0 \\ 0 & 0 & -\lambda \end{bmatrix},$$

$$C = C_i = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix} = \lambda I, 1 \leq i \leq n,$$
and \( I \) is a unit matrix of order two. In addition, these variables satisfy

\[
(A_0 + C_0) I = (A_1 + B_1 + C_1) I = \ldots = (A_n + B_n + C_n) I = (A + B + C) I = 0.
\]

Clearly, \( G \) has a block-tri-diagonal structure, which indicates that \( ((Q(t), V(t))) \) is a quasi-birth and death process. Let \( \rho_3 = \frac{\lambda}{\mu_3E} = \frac{\lambda}{\mu} \), and denote \( (Q, V) \) as the limit of \( (Q(t), V(t)) \). When \( \rho_3 < 1 \), denote the probability of \( (Q, V) \) in state \( (i, j) \in \Omega \), as \( p_{ij} \), the probability of \( i \) tasks as \( P_i \), when the TOC is busy and the probability of \( i \) tasks as \( p_i \), when the TOC is on vacation; i.e.,

\[
P_i = P\{Q = i, J = 0\}, \quad i \geq 1, \\
p_{ij} = P\{Q = i, J = j\}, \quad (i, j) \in \Omega, \\
p_i = p_{i1}, \quad i \geq 0.
\]

According to [33], the probability distribution of \( (Q, V) \) is obtained as

\[
P_i = C \left( \frac{\lambda}{\lambda + \delta} \right)^i, \quad i \geq 0, \\
P_i = C \left( \frac{\lambda}{\mu_3E} \right)^i \eta_i, \quad 1 \leq i \leq n - 1, \\
P_i = \rho_3 \rho_{n-1} \sum_{k=0}^{i-n-1} \rho_3^k \left( \frac{\lambda}{\lambda + \delta} \right)^{i-n-k} + \rho_3 \rho_{n-1} \sum_{k=0}^{i-n-1} \rho_3^k \left( \frac{\lambda}{\mu_3E} \right)^{i-n-k}, \quad i \geq n,
\]

where

\[
\eta_i = \sum_{k=0}^{i-1} k! \left( \frac{\mu_3E}{\lambda + \delta} \right)^k, \quad 1 \leq i \leq n - 1, \\
C = \left( \frac{\rho_3}{1 - \rho_3} \right) \left( \frac{\lambda}{\mu_3E} \right)^{n-1} \eta_{n-1} + \left( 1 - \frac{\rho_3}{1 - \rho_3} \right) \left( \frac{\lambda}{\lambda + \delta} \right)^{n-1} \left( 1 + \frac{\lambda}{\lambda - \delta} \right) + \sum_{j=1}^{n-1} \frac{1}{j!} \left( \frac{\lambda}{\mu_3E} \right)^j \eta_j.
\]

Then, the probability distribution of \( R_3 \) can be obtained when the scheduling queueing system is in equilibrium as

\[
P\{R_3 = 0\} = C, \quad P\{R_3 = j\} = P_j + p_j, \quad j \geq 1.
\]

Then

\[
R_3 = \sum_{i=1}^{n-1} C \left( \frac{\lambda}{\mu_3E} \right)^i \eta_i + P_{n-1} \frac{\alpha}{(1 - \rho_3)^2} \\
+ C \left( \frac{\lambda + \rho_3}{1 - \rho_3} \right) \left( \frac{\lambda}{\lambda + \delta} \right)^{n-1} \\
+ \left( \frac{\alpha \delta}{1 - \rho_3} \right) \left( \frac{\lambda}{\lambda + \delta} \right)^{n-1}.
\]

where \( \alpha = \rho_3 (\rho_3 n - n \rho_3) \). And

\[
T_3 = \frac{R_3}{\lambda}.
\]

Let \( q_0 \) and \( q_1 \) denote the steady-state probabilities of TOC being busy and on vacation, respectively; i.e.,

\[
q_0 = P\{J = 0\} = 1 - q_1, \\
q_1 = P\{J = 1\} = \sum_{k=0}^{\infty} P\{Q = k, J = 1\} = \sum_{k=0}^{\infty} p_{k} = C \sum_{k=0}^{\infty} \left( \frac{\lambda}{\lambda + \delta} \right)^k = C \left( 1 + \frac{\lambda}{\delta} \right).
\]

In addition, the mean utilization \( U \) of the OP is:

\[
U = \frac{1}{n} \sum_{i=1}^{\infty} i p_i, \quad (8)
\]

D. TRANSMITTING QUEUEING SYSTEM

As mentioned above, the mean arrival rate of tasks in the transmitting queueing system is also \( \lambda \). We also model the transmitting queueing system as an M/M/1 queueing system. The transmitting queueing system evenly processes and transmits \( D/2 \) for each task since the computing result of a binary tri-valued logic operation with \( M \) trits has \( M \) trits. Assume that the processing speed of the TM is equal to that of the PPM. That is, the speed of transforming the electrical signals into the final results is \( \tau \). Then, the service rate for transformation \( \mu_{41} = 2\tau/D \) and the service rate for transmission \( \mu_{42} = 2\xi/D \). When \( \mu_{41} = \lambda/\mu_{41} = \lambda D/2\tau < 1 \) and \( \mu_{42} = \lambda/\mu_{42} = \lambda D/2\xi < 1 \), the fourth queueing system has the same probability equations as the receiving queueing system. Therefore, the results in Section IV.A can be applied to \( R_4 \) and \( T_4 \).

\[
R_4 = \frac{\lambda}{2\tau - \lambda} + \frac{\lambda}{2\xi - \lambda}, \quad T_4 = \frac{1}{2\tau - \lambda} + \frac{1}{2\xi - \lambda}.
\]

The mean response time and the mean number of tasks in the system can be obtained by substituting (2)-(6) into (1).

VI. NUMERICAL VALIDATION AND SIMULATION

To validate the performance analysis model above, this section will explore the performance analysis and evaluation of the TOC through numerical examples and simulation experiments by using Python. We chiefly consider the influence on the system performance of the task arrival rate \( \lambda \), the number of small OPs \( n \) and the vacation rate \( \delta \). Certainly, the parameters are illustrative and can be altered to adapt to different TOC computing environments.

A. INFLUENCE ON THE SYSTEM PERFORMANCE OF THE TASK ARRIVAL RATE

To simulate the performance indicators, we first set the values of the parameters above. \( \lambda \in \{5i | 1 \leq i \leq 12, i \in N\} \) which indicates the mean number of tasks arriving per hour, \( \xi = 40MB/s, D = 2GB, \tau = 3GB/s, N = 10000, \sigma = 20GB/s, \)
TABLE 1. The results of $T_1$, $T_2$, $T_3$, $T_4$, $R_1$, $R_2$, $R_3$, $R_4$, $T$, and $R$ with different task arrival rates.

| $\lambda$ | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R$ | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 5  | 0.0746 | 0.0009 | 0.0140 | 0.0364 | 0.1260 | 53.7313 | 0.6673 | 10.0986 | 26.2328 | 90.7300 |
| 10 | 0.1613 | 0.0019 | 0.0280 | 0.0756 | 0.2667 | 58.0645 | 0.6679 | 10.0973 | 27.1993 | 96.0290 |
| 15 | 0.2632 | 0.0028 | 0.0421 | 0.1177 | 0.4257 | 63.1579 | 0.6685 | 10.0959 | 28.2408 | 102.1631 |
| 20 | 0.3846 | 0.0037 | 0.0561 | 0.1631 | 0.6076 | 69.2308 | 0.6691 | 10.0946 | 29.3662 | 109.3607 |
| 25 | 0.5319 | 0.0047 | 0.0701 | 0.2124 | 0.8191 | 76.5957 | 0.6698 | 10.0933 | 30.5862 | 117.9450 |
| 30 | 0.7143 | 0.0056 | 0.0841 | 0.2659 | 1.0699 | 85.7143 | 0.6704 | 10.0920 | 31.9132 | 128.3899 |
| 35 | 0.9459 | 0.0065 | 0.0981 | 0.3244 | 1.3749 | 97.2973 | 0.6710 | 10.0907 | 33.3619 | 141.4210 |
| 40 | 1.2500 | 0.0075 | 0.1121 | 0.3883 | 1.7579 | 112.5000 | 0.6716 | 10.0895 | 34.9500 | 158.2111 |
| 45 | 1.6667 | 0.0084 | 0.1261 | 0.4587 | 2.2599 | 133.3333 | 0.6723 | 10.0883 | 36.6984 | 180.7922 |
| 50 | 2.2727 | 0.0093 | 0.1401 | 0.5366 | 2.9587 | 163.6364 | 0.6729 | 10.0870 | 38.6328 | 213.0291 |
| 55 | 3.2353 | 0.0103 | 0.1541 | 0.6231 | 4.0228 | 211.7647 | 0.6735 | 10.0858 | 40.7845 | 263.3086 |
| 60 | 5.0000 | 0.0112 | 0.1681 | 0.7199 | 5.8992 | 300.0000 | 0.6742 | 10.0847 | 43.1923 | 353.9512 |

FIGURE 7. Mean requests and mean response times with different task arrival rates $\lambda$. (a) Mean requests in each stage with different task arrival rates $\lambda$; (b) Mean response time in each stage with different task arrival rates $\lambda$; (c) Mean requests in the whole system with different task arrival rates $\lambda$; (b) Mean response time in the whole system with different task arrival rates $\lambda$.

$\delta = 10$, $n = 5$. Let $\rho = \max\{\rho_1, \rho_2, \rho_3, \rho_41, \rho_42\}$. The system will be in equilibrium when $\rho < 1$. To compare the simulation results obtained by using the models above, the results of $T_1$, $T_2$, $T_3$, $T_4$, $R_1$, $R_2$, $R_3$, $R_4$, $T$, and $R$ are shown in Table 1 and Fig. 7.

Both the mean numbers of requests and the mean response times except for $T_3$ increase with the request arrival rate $\lambda$. Regardless of the value of $\lambda$, the receiving queueing system and the transmitting queueing system have the greatest influence on the mean of system requests $R$ and the average
FIGURE 8. Utilization and vacation probability of the TOC with different task arrival rates. (a) Utilization of the TOC with different task arrival rates $\lambda$; (b) Vacation probability of the TOC with different task arrival rates $\lambda$.

TABLE 2. The results of $R$, $T$, $U$, and $q_1$ with different values of $n$ and $\lambda$.

| $\lambda$ | $n=2$       |           | $n=3$       |           | $n=4$       |           | $n=5$       |           |
|-----------|--------------|-----------|--------------|-----------|--------------|-----------|--------------|-----------|
|           | $R$          | $T$       | $U$          | $q_1$     | $R$          | $T$       | $U$          | $q_1$     |
| 5         | 0.118        | 84.730    | 0.00278      | 0.994     | 0.120        | 86.731    | 0.00278      | 0.992     |
| 10        | 0.250        | 90.030    | 0.00556      | 0.989     | 0.256        | 92.030    | 0.00556      | 0.983     |
| 15        | 0.401        | 96.164    | 0.00833      | 0.983     | 0.409        | 98.165    | 0.00833      | 0.975     |
| 20        | 0.574        | 103.362   | 0.01111      | 0.978     | 0.585        | 105.363   | 0.01111      | 0.967     |
| 25        | 0.777        | 111.947   | 0.01389      | 0.973     | 0.791        | 113.948   | 0.01389      | 0.959     |
| 30        | 1.020        | 122.393   | 0.01667      | 0.967     | 1.037        | 124.393   | 0.01667      | 0.951     |
| 35        | 1.317        | 135.424   | 0.01945      | 0.962     | 1.336        | 137.425   | 0.01944      | 0.943     |
| 40        | 1.691        | 152.215   | 0.02223      | 0.957     | 1.714        | 154.215   | 0.02222      | 0.936     |
| 45        | 2.185        | 174.797   | 0.02502      | 0.951     | 2.210        | 176.797   | 0.02500      | 0.928     |
| 50        | 2.875        | 207.035   | 0.02780      | 0.946     | 2.903        | 209.034   | 0.02778      | 0.920     |
| 55        | 3.931        | 257.315   | 0.03059      | 0.941     | 3.962        | 259.314   | 0.03056      | 0.912     |
| 60        | 5.799        | 347.959   | 0.03337      | 0.936     | 5.833        | 349.957   | 0.03334      | 0.905     |

| $\lambda$ | $\delta = 0.01$ |           | $\delta = 0.1$ |           | $\delta = 1$ |           | $\delta = 10$ |           |
|-----------|-----------------|-----------|-----------------|-----------|--------------|-----------|-----------------|-----------|
|           | $R$             | $T$       | $U$             | $q_1$    | $R$          | $T$       | $U$             | $q_1$    |
| 5         | 0.261           | 187.583   | 0.003           | 0.990    | 0.137        | 98.522    | 0.003           | 0.989    |
| 10        | 0.533           | 191.925   | 0.006           | 0.981    | 0.288        | 103.715   | 0.006           | 0.978    |
| 15        | 0.822           | 197.179   | 0.008           | 0.973    | 0.457        | 109.745   | 0.008           | 0.968    |
| 20        | 1.131           | 203.579   | 0.011           | 0.966    | 0.649        | 116.842   | 0.011           | 0.958    |
| 25        | 1.468           | 211.451   | 0.014           | 0.960    | 0.870        | 125.327   | 0.014           | 0.948    |
| 30        | 1.844           | 221.267   | 0.017           | 0.955    | 1.131        | 135.674   | 0.017           | 0.938    |
| 35        | 2.273           | 233.750   | 0.020           | 0.951    | 1.445        | 148.609   | 0.019           | 0.928    |
| 40        | 2.779           | 250.068   | 0.023           | 0.947    | 1.837        | 165.306   | 0.022           | 0.919    |
| 45        | 3.403           | 272.248   | 0.027           | 0.944    | 2.347        | 187.795   | 0.025           | 0.910    |
| 50        | 4.224           | 304.149   | 0.030           | 0.941    | 3.055        | 219.941   | 0.028           | 0.901    |
| 55        | 5.411           | 354.152   | 0.034           | 0.938    | 4.127        | 270.131   | 0.031           | 0.892    |
| 60        | 7.410           | 444.573   | 0.037           | 0.936    | 6.011        | 360.686   | 0.033           | 0.884    |
system response time $T$. The reason is that the network transmission speed $\xi$ is too small compared with the speed of the master computer $\tau$ and the speed of the TOC $\sigma$, which causes the service rate of the two queueing systems to be low. $T_3$ is particularly interesting and important. $T_3$ as shown in Table 1 demonstrates little change and a decreasing trend, although $R_3$ increases with the request arrival rate $\lambda$.

The utilization of the TOC $U$, as shown in Fig. 8(a), also increases with the task arrival rate $\lambda$, and $U$ is also low when $\lambda$ is low. The reason is that the vacation probability of the TOC $q_1$ is high when $\lambda$ is low, as shown in Fig. 8(b); only a few small OPs are occupied and the others are idle.

In short, $R$, $T$, and $U$ all increase and $q_1$ decreases with the increase of $\lambda$. This demonstrates that they are in agreement with what we expect, i.e., they can be trusted.

**B. INFLUENCE ON THE SYSTEM PERFORMANCE OF THE NUMBER OF SMALL OPTICAL PROCESSORS**

In this section, we will discuss the influence on the system performance, especially $R$, $T$, $U$, $q_1$, of the number of small OPs $n$, where $n \in \{2, 3, 4, 5\}$. When the other parameters are unchanged, the results are shown in Table 2 and Fig. 9. We can observe that $R$ and $T$ both increase with the increase of $n$. The reason is that $U$ and $q_1$ both decrease with the increase of $n$. The difference between them is that the decrease in $U$ is tiny, and is difficult to find in Fig. 9(c), while the decrease in $q_1$ is remarkable, as shown in Fig. 9(d).

As expected, the increase of $n$ leads to a decrease in system performance. Consequently, we should use the optical processor as a whole for the improvement of the TOC under the PDE strategy.

**C. INFLUENCE ON THE SYSTEM PERFORMANCE OF THE VACATION RATE**

In this section, we will discuss in detail the impact of the vacation rate $\delta$ on system performance, where $\delta \in \{0.01, 0.1, 1, 10\}$ and $n = 4$. The results are shown in Table 3 and Fig. 10. It is particularly interesting that $R$, $T$, $U$, and $q_1$ do not decrease with decreasing vacation rate. In particular, these performance indicators, especially the response time, increase significantly when $\delta = 0.01$. The reason is that the smaller $\delta$ is, the longer the vacation time is. In other words, the high vacation rate can improve the system performance to some degree.
VII. CONCLUSIONS AND FUTURE WORK

Performance analysis and evaluation is a significant aspect of a TOC for both its providers and its customers. On the basis of an analysis of the characteristics of a TOC and its service processes, we proposed a complex queueing model composed of four concatenated queueing systems—the receiving queueing system, the preprocessing queueing system, the scheduling queueing system, and the transmitting queueing system—to evaluate the performance of TOC. Additionally, we introduced synchronous multi-vacations into the scheduling queueing system and the transmitting queueing system—to evaluate the performance of the TOC. We chose some primary performance indicators, such as the mean response time and the mean number of requests in the system, and analyzed them theoretically. Finally, we also conducted simulation experiments to validate the complex queueing model. The results showed that the proposed model allowed a sophisticated analysis of the TOC. Moreover, we conducted experiments to analyze key factors, such as the number of small optical processors and the vacation rate, that impact the performance of TOC. Based on the complex queueing model, we plan to extend our analytical model by applying asynchronous multiple vacations to optimize the performance of TOC.

REFERENCES

[1] F. Zangeneh-Nejad, A. Khavasi, and B. Rejaei, “Analog optical computing by half-wavelength slabs,” Opt. Commun., vol. 407, pp. 338–343, Jan. 2018, doi: 10.1016/j.optcom.2017.09.078.
[2] H. Babashah, Z. Kavehvash, A. Khavasi, and S. Koohi, “Temporal analog optical computing using an on-chip fully reconfigurable photonic signal processor,” Opt. Laser Technol., vol. 111, pp. 66–74, Apr. 2019, doi: 10.1016/j.optlastec.2018.09.027.
[3] T. Yue, J. Suo, and Q. Dai, “Non-uniform image deblurring using an optical computing system,” Comput. Graph., vol. 37, no. 8, pp. 1039–1050, Dec. 2013, doi: 10.1016/j.cag.2013.10.011.
[4] J. Pang, A. R. Lebeck, and C. Dwyer, “Modeling and simulation of a nanoscale optical computing system,” J. Parallel Distrib. Comput., vol. 74, no. 6, pp. 2470–2483, Jun. 2014, doi: 10.1016/j.jpdc.2013.07.006.
[5] Y. Jin, H. C. He, and Y. T. Lü, “Ternary optical computer principle,” Sci. China F Inform. Sci., vol. 46, no. 2, pp. 145–150, Apr. 2003, doi: 10.1360/03y09012.
[6] Y. Jin, H. C. He, and Y. T. Lü, “Ternary optical computer architecture,” Phys. Scripta, vol. 118, pp. 98–101, Feb. 2005.
[7] J. Yan, Y. Jin, and K. Zuo, “Decrease-radix design principle for carrying/borrowing free multi-valued and application in ternary optical computer,” Sci. China Ser. F, Inf. Sci., vol. 51, no. 10, pp. 1415–1426, Oct. 2008, doi: 10.1007/s11432-008-0140-x.
[8] X. Wang, J. Peng, M. Li, Z. Shen, and O. Shan, “Carry-free vector-matrix multiplication on a dynamically reconfigurable optical platform,” Appl. Opt., vol. 49, no. 12, pp. 2352–2362, Apr. 2010, doi: 10.1364/AO.49.002352.
Y. Jin, Y. Shen, J. Peng, S. Xu, G. Ding, D. Yue, and H. You, “Principles and construction of MSD adder in ternary optical computer,” Sci. China Inf. Sci., vol. 53, no. 11, pp. 2159–2168, Nov. 2010, doi: 10.1007/s11432-010-4091-9.

J. Peng, R. Shen, Y. Jin, Y. Shen, and S. Luo, “Design and implementation of modified signed-digit adder,” IEEE Trans. Comput., vol. 63, no. 5, pp. 1134–1143, May 2014.

K. Song and L. Yan, “Design and implementation of the one-step MSD adder of optical computer,” Appl. Opt., vol. 51, no. 7, pp. 917–926, Mar. 2012, doi: 10.1364/AO.51.000917.

S. Ouyang, J. J. Peng, Y. Jin, Y. F. Shen, X. M. Liu, and W. M. Li, “Structure and theory of dual-space storage for ternary optical computer,” Sci. China Inform. Sci., vol. 46, no. 6, pp. 743–762, Jun. 2016, doi: 10.1007/s11432-015-00036.

Q. Xu, X. Wang, and C. Xu, “Design and implementation of the modified signed digit multiplication routine on a ternary optical computer,” Appl. Opt., vol. 56, no. 16, pp. 4661–4669, Jun. 2017, doi: 10.1364/AO.56.004661.

S. L. Zhang, J. J. Peng, Y. F. Shen, and X. C. Wang, “Programming model and implementation mechanism for ternary optical computer,” Opt. Commun., vol. 428, no. 1, pp. 26–34, Jan. 2018, doi: 10.1016/j.optcom.2018.07.038.

S. Kai and Y. Liping, “Control mechanism of double-rotator-structure ternary optical computer,” Opt. Commun., vol. 387, pp. 338–349, Mar. 2017, doi: 10.1016/j.optcom.2016.12.004.

X. C. Wang, S. L. Zhang, M. Zhang, J. Zhao, and X. Y. Niu, “Performance analysis of a ternary optical computer based on M/M/1 queueing system,” in Proc. ICAAPP, Helsinki, Finland, 2017, pp. 331–344.

Q. Xu and X. Wang, “Service model and performance analysis of ternary optical computer based on complex queueing system,” J. Nat. Univ. Defense Technol., vol. 39, no. 2, pp. 140–145, Apr. 2017, doi: 10.1187/jnudt.201702021.

X. C. Wang, S. L. Zhang, S. Gao, M. Zhang, J. Zhang, X. C. Wang, and Z. Xu, “Response time of a ternary optical computer that is based on queueing systems,” J. Supercomput., vol. 2019, pp. 1–20, Feb. 2019, doi: 10.1007/s11227-019-02771-3.

X. C. Wang, J. Zhang, S. Gao, G. Sun, J. Zhao, M. Zhang, L. G. Li, D. D. Liu, Y. Zhang, and X. C. Wang, “Response time of a ternary optical computer based on complex queueing system with synchronous multi-vacations,” in Proc. ICTACSI, Huainan, China, 2019, pp. 1180–1188.

K. K. Starkov, A. Y. Pogromsky, and J. E. Rooda, “Performance analysis for tandem manufacturing lines under variable structure production control method,” Int. J. Prod. Res., vol. 50, no. 8, pp. 2363–2375, Apr. 2012.

J. D. Deng and M. K. Purvis, “Multi-core application performance optimization using a constrained tandem queueing model,” J. Netw. Comput. Appl., vol. 34, no. 6, pp. 1990–1996, Nov. 2011, doi: 10.1016/j.jnca.2011.07.004.

I. F. Akdyldiz, “Mean value analysis for blocking queueing networks,” IEEE Trans. Softw. Eng., vol. 14, no. 4, pp. 418–428, Sep. 1988, doi: 10.1109/32.4663.

G. N. Satish, C. V. Raghavendra, M. V. R. Sundari, and P. S. Varma, “Performance evaluation of three node tandem communication network model with feedback for first two nodes having homogeneous Poisson arrivals,” Int. J. Comput. Sci., vol. 98, no. 12, pp. 26–36, Jul. 2014.

N. T. Rao, K. S. Rao, P. S. Rao, and K. N. Rao, “Performance evaluation of two node tandem communication network with DBA having compound Poisson binomial bulk arrivals,” Comput. Eng. Intell. Syst., vol. 5, no. 1, pp. 14–17, Jan. 2014.

M. Mitici, J. Goseling, J. K. van Ommeren, M. de Graaf, and R. J. Boucherie, “On a tandem queue with batch service and its applications in wireless sensor networks,” Queueing Syst., vol. 87, nos. 1–2, pp. 81–93, Oct. 2017, doi: 10.1007/s11134-017-9534-1.

K. Sigman, “Regeneration in tandem queues with multiserver stations,” J. Appl. Probab., vol. 25, no. 2, pp. 391–405, Jun. 1988, doi: 10.2307/3214447.

T. T. Lee, “M/G/1/NQueue with vacation time and exhaustive service discipline,” Operations Res., vol. 32, no. 4, pp. 774–784, Aug. 1984, doi: 10.1287/opre.32.4.774.

A. Frey and Y. Takahashi, “An explicit solution for an M/GI/1 queue with vacation time and exhaustive service discipline,” J. Oper. Res. Soc. Jpn., vol. 41, no. 3, pp. 430–441, 1998, doi: 10.1887/jsorj.41.430.

Y. S. Cheng and Y. J. Zhu, “Analysis of departure structure of M/GI/1 type vacation queue with nonexhaustive service,” J. Jiangsu Univ. Nat. Sci., vol. 25, no. 3, pp. 239–242, May 2004.
SHAN GAO received the Ph.D. degree from Central South University, Changsha, China, in 2011. She is currently a Professor with the Department of Mathematics and Statistics, Fuyang Normal University. Her research interests include queueing theory, stochastic process, and risk theory. She has published over 20 research articles in these areas. She is a member of the Operations Research Society of China.

MIAN ZHANG received the Ph.D. degree from Central South University, Changsha, China, in 2011. She is currently a Professor with Fuyang Normal University, China. Her research focuses on queueing modeling and theory. She is a member of the Operations Research Society of China.

XIANCHAO WANG was born in Suzhou, Anhui, China, in 1973. He received the B.S. degree in mathematics education from Fuyang Normal University, China, in 1997, the master’s degree in computer science from Northeastern University, China, in 2004, and the Ph.D. degree in computer science from Shanghai University, Shanghai, China, in 2011. From 2010 to 2016, he was an Assistant Professor with the School of Mathematics and Statistics, Fuyang Normal University, China, where he has been a Professor with the School of Computer and Information Engineering, since 2017. He has coauthored over 30 conference publications, journal articles, and a book and holds three patents. His research interests include queueing modeling and theory, computer software, optical computing, complex networks, big data, and embedded systems. He is a member of the ACM and a Senior Member of China Computer Federation.