Journal of Parallel and Distributed Computing 130 (2019) 24–36

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Optimal power allocation and load balancing for non-dedicated heterogeneous distributed embedded computing systems

Jing Huang [a,b], Yan Liu [a,b,*], Renfa Li [a,b,**], Keqin Li [a,c], JiYao An [a,b], Yang Bai [a,b], Fan Yang [d], Guoqi Xie [a,b]

[a] College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan, 410082, China
[b] Key Laboratory for Embedded and Network Computing of Hunan Province, Hunan University, Changsha, Hunan, 410082, China
[c] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA
[d] College of Computer and Information Engineering, Central South University of Forestry and Technology, Changsha, Hunan, 410004, China

HIGHLIGHTS

- The considered system is fully heterogeneous.
- Task types and priorities are diverse.
- Two new rules are observed from a great deal of data experiments.
- An optimal power allocation and load balancing strategy is proposed.

ARTICLE INFO

Article history:
Received 29 March 2017
Received in revised form 28 July 2018
Accepted 23 March 2019
Available online 29 March 2019

MSC:
00-01
99-00

Keywords:
Embedded and distributed system
Load distribution
Power allocation
Queueing model
Response time

ABSTRACT

This paper investigates on the optimal power allocation and load balancing problem encountered by heterogeneous and distributed embedded systems with mixed tasks. Given that each node has real and different urgent tasks in the majority of practical heterogeneous embedded systems, three priority disciplines are considered: dedicated jobs without priority, prioritized dedicated jobs without preemption, and prioritized dedicated jobs with preemption. A model is established for heterogeneous embedded processors with dedicated-task-dependent dynamic power and load balancing management; each processor is considered as an M/M/1 queueing sub-model with mixed generic and dedicated tasks. The processors have different levels of power consumption, and each one can employ any of the three disciplines. The objective of this study is to find an optimal load balancing (for generic tasks) and power allocation strategy for heterogeneous processors preloaded by different amounts of dedicated tasks such that the average response time of generic tasks is minimized. Considering that this problem is a multi-constrained, multi-variable optimization problem for which a closed-form solution is unlikely to be obtained, we propose an optimal power allocation and load balancing scheme by employing Lagrange method and binary search approach, which are completed by utilizing two new rules established by observing numerical variations of parameters. Several numerical examples are presented to demonstrate the effectiveness of our solution. To the best of our knowledge, this is the first work on analytical study that combines load balancing, energy efficiency, and priority of tasks in heterogeneous and distributed embedded systems.

1. Introduction

1.1. Motivation

In view of the increasing computing demands and performance requirements of modern applications, complex embedded platforms are now equipped with multiple computing nodes, in which each node owns its dedicated jobs. These nodes vary in computing speed and power consumption. As such, embedded platforms exhibit characteristics of heterogeneous computing, embedded computing, and distributed computing. Typical

* Corresponding author at: College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan, 410082, China.
** Corresponding author.
E-mail addresses: jingh@hnu.edu.cn (J. Huang), liuyan@hnu.edu.cn (Y. Liu), lirenfa@vip.sina.com.cn (R. Li), lik@newpaltz.edu (K. Li), anbobcn@aliyun.com (J. An), baiyang@hnu.edu.cn (Y. Bai), yangfanf117@csuft.edu.cn (F. Yang), xgqman@hnu.edu.cn (G. Xie).

https://doi.org/10.1016/j.jpdc.2019.03.019
0743-7315/

examples of embedded platforms are the BMW 7 Series cars, in which over 100 electronic control units (ECUs) are embedded in a single car with dedicated tasks for each ECU.

Computing performance is a vital metric when a system's quality of service (QoS) is being evaluated. Higher computing performance results in a shorter average waiting time for tasks and a higher system throughput. High computing performance depends on the effective utilization of computing resources. Therefore, assigning generic tasks to nodes without affecting their own dedicated tasks is an effective strategy for performance improvement of embedded systems. In distributed systems, task allocation assigned by a load-balancing algorithm is directly related to the overall performance of the entire system. Thus, an efficient load-balancing strategy is vital to building a distributed-computing architecture, particularly an embedded distributed system with limited resource.

Energy efficiency is important in most computing environments. Embedded systems depend even more heavily on energy efficiency and optimization than general systems. Energy efficiency is related to the performance and runtime of a system, especially for battery-powered devices. Energy management can be achieved by using dynamic power management (DPM) [19] or dynamic voltage scaling (DVS) [32] to regulate power consumption. DPM implemented at the operating-system level provides the supply voltage and clock-frequency adjustment schemes while tasks are running. This strategy offers opportunities for tuning the energy-delay tradeoff [27].

In distributed embedded systems, each node owns its dedicated tasks, which may be urgent tasks for the system. Thus, in such environments, the priority of dedicated tasks should be considered when tasks are being assigned to a node. The urgency of the dedicated tasks of each node is not the same. Thus, the priority strategy could be set in various ways. That is, some dedicated tasks can be considered concomitant general tasks, some tasks possess a higher priority than general tasks, and some tasks possess a higher priority with pre-emption.

With the development of embedded computing, heterogeneous and multi-node architectures have become a trend. In heterogeneous distributed embedded environments with limited computing resource and power, a system should be capable of operating with optimal load balancing and power allocation without impacting the system's important tasks.

1.2. Our contributions

In this paper, we aim to develop power- and performance-constrained load-distribution methods for current and future embedded architectures. We examine the problem of assigning a set of general tasks to computing nodes of a computational heterogeneous embedded distributed system, wherein each node is pre-loaded with different amounts of dedicated tasks and is equipped with a DVS feature. However, each dedicated task for each node has a different priority because of the differences in urgency to the system. According to the urgency of the dedicated tasks, three types of priority disciplines of nodes in the system are as follows:

- Discipline 1: All general tasks and dedicated tasks on this node are scheduled with a first-come, first-served discipline without priority.
- Discipline 2: Dedicated tasks are always scheduled before general tasks on this node. All tasks are executed without interruption.
- Discipline 3: Dedicated tasks are always scheduled before general tasks on this node and with pre-emption.

We refer to Discipline 1 as dedicated tasks without priority, Discipline 2 as prioritized dedicated tasks without pre-emption, and Discipline 3 as prioritized dedicated tasks with pre-emption. Each node may employ any of the three disciplines. We treated each embedded computing node as an M/M/1 queueing model with infinite waiting-queue capacity.

The contributions of this study are as follows:

- An algorithm is proposed for finding the optimal load-distribution and power-allocation schemes of the system, such that the overall average response time for generic tasks is minimized.
- The proposed algorithm is highly adaptable, such that it can be used to deal with heterogeneous nodes with different amounts of pre-loaded dedicated tasks, different queueing disciplines, as well as different maximum speeds and power consumption.
- We provide several numerical examples to demonstrate the effectiveness of our algorithm from different perspectives. We also plot the functions of overall response time of generic tasks, node utilization of each node, power allocation on each node, and task assignment on each node of the arrival rate of generic tasks. Through these functions, we can observe the changing tendency of these parameters as the arrival rate of generic tasks changes.

Our study is focused on a well-defined, multi-constrained, and multi-variable problem. To the best of our knowledge, no study has investigated an optimization problem that considers the combination of load distribution, energy efficiency, and task priority in heterogeneous embedded distributed systems. Our investigation makes a significant contribution to high-performance and energy-efficient computing in modern heterogeneous and distributed embedded systems.

2. Related work

Load distribution and balancing in general distributed and parallel computing systems have been extensively studied, and a huge body of literature exists [3,15,25,29]. The majority of existing works on optimal load distribution have focused on performance metrics and system characteristics. Performance metrics include mean response time [7], arithmetic average response time [13], average consensus [34], mean response ratio [28], mean miss ration [9], and probability of load-balancing success [21]. Distributed-system characteristics refer to two main considerations: (1) whether the server is heterogeneous, and (2) whether each server has its own dedicated jobs and whether these jobs have higher priority. In Ref. [36], the authors studied how to quickly find the minimum response time of an application running on a heterogeneous system. In Ref. [35], the adaptive dynamic scheduling on heterogeneous embedded systems was investigated. In Ref. [6], the system contains non-priority dedicated jobs and generic jobs, and the objective of this paper is to seek assignment probabilities of generic jobs such that the average response time of a job is minimized. In Ref. [22], the author considered a heterogeneous environment that includes general jobs and multiple types of non-priority dedicated jobs; this work explored the problem of optimal job-dispatching probabilities for minimum average job response time. In Ref. [12], three types of priorities, namely dedicated jobs without priority, priority dedicated jobs without pre-emption, and priority dedicated jobs with pre-emption were considered in a heterogeneous environment; however, in Ref. [12], all of the priority disciplines of each node are set to be the same and not heterogeneous. In Ref. [18], the quantitative modeling and analytical calculation of elasticity in distributed systems was studied. In Ref. [17], optimal task

dispatching on multiple heterogeneous multi-server systems was studied. Virtually, the nature of problems discussed in Refs. [18] and [17] are still load balancing.

Energy efficiency is necessary in both distributed (e.g., cloud computing, grid computing) or embedded environments. For large-scale distributed systems, high energy consumption leads to severe economic, environmental, and ecological problems. For embedded systems, high energy consumption will affect the system's runtime, particularly that of battery-power devices. The intuitionistic policy for reducing power consumption is to efficiently manage system power. In Ref. [11], the author investigated the green-energy-aware power-management problem for distributed-computing data centers. In Ref. [8], the author summarized and classified resource-management problems in distributed-computing environments. In Ref. [20], the author summarized the approaches and techniques proposed in recent works in this discipline. The purpose of energy efficiency is to make power consumption proportional to system utilization [4]. Energy efficiency can be achieved by applying many different approaches, such as algorithmic, stochastic, and learning-based approaches [1]. In Ref. [26], the author developed a methodology for cycle-accurate simulation of performance and energy consumption in embedded systems. In Ref. [39], a framework was built for designing and analyzing the energy cost of algorithms. Basic software techniques for energy efficiency are supported by a mechanism called dynamic frequency scaling (equivalently, DVS, dynamic speed scaling, and dynamic power scaling). DPM at the operating-system level provides supply-voltage and clock-frequency adjustment schemes implemented while tasks are running. A huge body of literature is available on this topic. In Ref. [5], the author summarized several classes of power-managed systems and power-management strategies. These energy-saving techniques explore the opportunities for tuning the energy-delay tradeoff. In Ref. [23], the researcher specifically studied the problem of system-level design techniques for the energy efficiency of embedded systems.

Both performance and energy efficiency are important for most systems, especially for distributed embedded systems. In general, three policies can be implemented to combine DPM and performance to achieve energy efficiency. The first is to optimize the performance by fixing the energy consumption constraint. In Ref. [7], the author addressed the problem of minimizing average response time of tasks under a given power. In Ref. [14], the author minimized the average task response time of multiple heterogeneous servers under the condition that the overall power consumption of all servers does not exceed a pre-set power limit. In Ref. [16], the author studied the problem of finding the optimal speed scheme of servers with workload-dependent DPM to minimize the average task response time under a given power. The second consideration is to minimize the power consumption under a performance constraint [30,31,37,39]. The author of Ref. [30] proposed a method that seeks the minimum energy consumption of a specific task set by using resource sharing that meets the time constraint. In Ref. [31], the author used dynamic voltage–frequency scaling to minimize power consumption for precedence-constrained parallel task execution without increasing the entire task's execution time. The third consideration is to optimize both power consumption and performance [10,33]. In Ref. [10], the energy consumption and the make span are minimized without violating the deadlines and the tasks' architectural requirements. In Ref. [33], the author minimized the weighted sum of mean response time and power consumption while scheduling processor sharing.

Table 1
Notations.

s_i	The speed of node v_i
$\widetilde{\lambda}_i$	Arrival rate of dedicated tasks to v_i
$\widehat{\lambda}_i$	Arrival rate of general tasks to v_i
λ_i	$= \widetilde{\lambda}_i + \widehat{\lambda}_i$
$\widehat{\lambda}$	$= \sum_{i=1}^{n} \widehat{\lambda}_i$
$\widetilde{x}_i = \widetilde{r}_i/s_i$	Average execution time of dedicated tasks on v_i
$\widehat{x}_i = \widehat{r}/s_i$	Average execution time of general tasks on v_i
$\widetilde{u}_i = 1/\widetilde{x}_i$	Average service rate of dedicated tasks on v_i
$\widehat{u}_i = 1/\widehat{x}_i$	Average service rate of general tasks on v_i
$\widetilde{m}_i = 2/\widetilde{u}_i^2$	The second moment of dedicated tasks on v_i
$\widehat{m}_i = 2/\widehat{u}_i^2$	The second moment of general tasks on v_i
m_i	$= \left(\widetilde{\lambda}_i/\lambda_i\right) \cdot \widetilde{m}_i + \left(\widehat{\lambda}_i/\lambda_i\right) \cdot \widehat{m}_i$
$\widehat{\rho}_i$	$\widehat{\lambda}_i \cdot \widehat{x}_i = \widehat{\lambda}_i \widehat{r}/s_i$
$\widetilde{\rho}_i$	$\widetilde{\lambda}_i \cdot \widetilde{x}_i = \widetilde{\lambda}_i \widetilde{r}_i/s_i$
$\rho_i = \widehat{\rho}_i + \widetilde{\rho}_i$	Average percentage of time that node v_i is busy
T_i	Average response time of general tasks on v_i
T	Average response time of general tasks on system

3. Models and problem formulation

3.1. Power model

The power dissipation of CMOS circuits mainly consists of three parts, namely, dynamic, static, and short-circuits consumption, among which dynamic power consumption is the dominant component. The dynamic power consumption can be expressed by $P = kCV^2f$ where k is an activity factor, C is the loading capacitance, V is the supply voltage, and f is the clock frequency. The speed s of a core is often defined as the number of instructions the core can perform per second (IPS). Given that $s \propto f$ and $f \propto V$, then $P \propto s^{\alpha}$, where α is around 3 [38]. For ease of discussion, we model the dynamic power allocated to node v_i with speed s_i as ${s_i}^{\alpha_i}$. Thus, we formulate the power consumption of an embedded computing node as $P_i = {s_i}^{\alpha_i} + {P_i}^*$, where ${P_i}^*$ is the static power consumption independent of the clock rate.

3.2. Queueing model

We establish a model to formulate and study the problem of power allocation and load balancing in a heterogeneous distributed embedded environment. We assume that we have n heterogeneous embedded computing nodes $v_1, v_2, \ldots, v_n$ (simply called as a node), each of which has its own dedicated set of jobs, which is a Poisson stream of tasks with arrival rate $\widetilde{\lambda}_i$ that can only be executed on it. There exists a general Poisson stream of tasks with arrival rate $\widehat{\lambda}$ that needs to be executed by being split into n sub-streams $\widehat{\lambda}_i$ assigned to each node. Thus, each node deals with a combined stream of dedicated and general tasks. The structure of the system is showed in Fig. 1. The task execution requirements of dedicated and general tasks are exponential random variables with mean $\widetilde{r}_i$ and $\widehat{r}$, respectively. Thus, the two types of mean execution times on node v_i are $\widetilde{x}_i = \widetilde{r}_i/s_i, \widehat{x}_i = \widehat{r}/s_i$, respectively. Since both the arrival rate and processing rate of tasks submit to Poisson distribution, each node can be treated as an M/M/1 queueing system. Parameters used are shown in Table 1. To maintain the queue steady, we assume that $\rho_i < 1$, for all $1 \le i \le n$.

3.3. Problem formulation

Our problem can be specified as follows: given n numbers of embedded nodes/cores $v_1, v_2, \ldots, v_n$, the arrival rates $\widetilde{\lambda}_1, \widetilde{\lambda}_2, \ldots,$

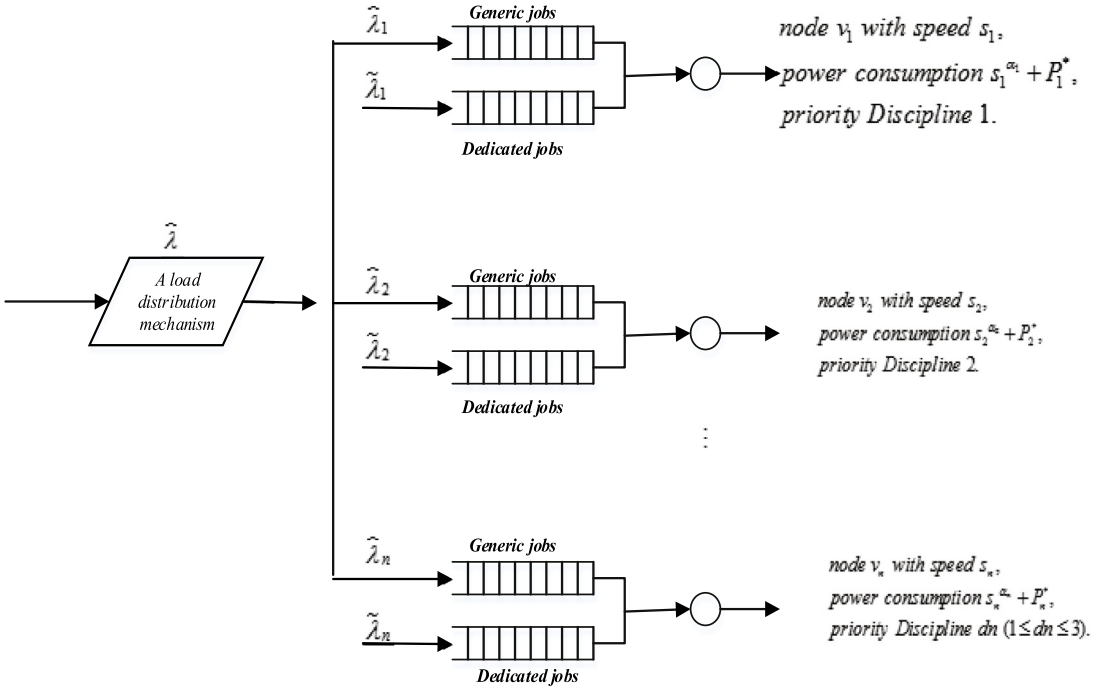

Fig. 1. System structure.

$\widetilde{\lambda}_n$ of dedicated tasks on the n nodes, the average tasks execution requirements $\widetilde{r}_1, \widetilde{r}_2, \ldots, \widetilde{r}_n$ of dedicated tasks on the n nodes, the total arrival rate $\widehat{\lambda}$ of general tasks, the average task execution requirements $\widehat{r}$ of general tasks, the base power supply $P_1^*, P_2^*, \ldots, P_n^*$, the available power $\overline{P}$, and the queueing discipline of each node, find the tasks arrival rates $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n$ and the core speeds $s_1, s_2, \ldots, s_n$ such that the average response time of general tasks on the system is minimized, namely,

$$T = \frac{\widehat{\lambda}_1}{\widehat{\lambda}} T_1 + \frac{\widehat{\lambda}_2}{\widehat{\lambda}} T_2 + \cdots + \frac{\widehat{\lambda}_n}{\widehat{\lambda}} T_n, \tag{1}$$

subject to

$$\widehat{\lambda}_1 + \widehat{\lambda}_2 + \cdots + \widehat{\lambda}_n = \widehat{\lambda}, \tag{2}$$

and

$$\overline{P} = \sum_{i=1}^{n} P_i = \sum_{i=1}^{n} (s_i^{\alpha_i} + P_i^*), \tag{3}$$

where T_i $(1 \le i \le n)$ represents the average response time of generic tasks on v_i.

4. The proposed method

4.1. Basic derivation

Each node is treated as an M/M/1 queuing system. For different queuing disciplines, the average response time of general tasks T_i is different. We split all nodes into three groups according to the queuing discipline. Group G_1 includes all of the nodes whose priority disciplines are Discipline 1, group G_2 includes all of the nodes whose priority disciplines are Discipline 2, and group G_3 includes all of the nodes whose priority disciplines are Discipline 3. If $v_i \in G_1$, we have [[2], p. 700]

$$\begin{aligned} T_i &= \widehat{x}_i + \frac{\lambda_i m_i}{2(1-\rho_i)} \\ &= \frac{\widehat{r}}{s_i} + \frac{\widehat{\lambda}_i \widehat{r}^2 + \widetilde{\lambda}_i \widetilde{r}_i^2}{s_i \left(s_i - \widehat{\lambda}_i \widehat{r} - \widetilde{\lambda}_i \widetilde{r}_i\right)}. \end{aligned} \tag{4}$$

If $v_i \in G_2$, we have [[2], p. 702]

$$\begin{aligned} T_i &= \widehat{x}_i + \frac{\lambda_i m_i}{2(1-\widetilde{\rho}_i)(1-\rho_i)} \\ &= \frac{\widehat{r}}{s_i} + \frac{\widehat{\lambda}_i \widehat{r}^2 + \widetilde{\lambda}_i \widetilde{r}_i^2}{\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i\right)\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i - \widehat{\lambda}_i \widehat{r}\right)}. \end{aligned} \tag{5}$$

If $v_i \in G_3$, we have [[2], p. 704]

$$\begin{aligned} T_i &= \frac{1}{1-\widetilde{\rho}_i}\left(\widehat{x}_i + \frac{\lambda_i m_i}{2(1-\rho_i)}\right) \\ &= \frac{1}{s_i - \widetilde{\lambda}_i \widetilde{r}_i}\left(\widehat{r} + \frac{\widehat{\lambda}_i \widehat{r}^2 + \widetilde{\lambda}_i \widetilde{r}_i^2}{s_i - \widetilde{\lambda}_i \widetilde{r}_i - \widehat{\lambda}_i \widehat{r}}\right). \end{aligned} \tag{6}$$

Our target is to minimize Eq. (1). Note that $\widehat{\lambda}$ in Eq. (1) is a fixed value, it could be ignored that would not impact the original solution. To solve Eq. (1), we use the Lagrange multiplier system, and set the following functions:

$$\psi(s_1, s_2, \ldots, s_n) = P - \sum_{i=1}^{n} P_i,$$

and

$$\varphi(\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n) = \sum_{i=1}^{n} \widehat{\lambda}_i - \widehat{\lambda},$$

as two Lagrange constraint functions, and construct a Lagrange function:

$$L = \sum_{i=1}^{n} \widehat{\lambda}_i T_i + \phi \psi(s_1, s_2, \ldots, s_n) + \tau \varphi(\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n), \tag{7}$$

where ϕ and τ are two Lagrange multipliers.

Eq. (7) includes 2n + 2 variables, which are $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n$, $s_1, s_2, \ldots, s_n, \phi, \tau$. We take the partial derivative with respect to

$\widehat{\lambda}_i$, which is

$$\frac{\partial\left(\widehat{\lambda}_i T_i\right)}{\partial\widehat{\lambda}_i} = \phi\frac{\partial\varphi}{\partial\widehat{\lambda}_i} + \tau\frac{\partial\psi}{\partial\widehat{\lambda}_i}. \tag{8}$$

According to Eqs. (1) and (8), we have

$$T_i + \frac{\partial T_i}{\partial\widehat{\lambda}_i}\widehat{\lambda}_i = \phi. \tag{9}$$

In Eq. (9), T_i and $\frac{\partial T_i}{\partial\widehat{\lambda}_i}$ vary under different priority disciplines. Hence, $\widehat{\lambda}_i$ should be solved according to the priority discipline of v_i.

For $v_i \in G_1$, based on Eqs. (4) and (9), we have

$$T_i + \left(\frac{\widehat{r}^2\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i\right) + \widetilde{\lambda}_i\widehat{r}^2\widehat{r}}{s_i\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i - \widehat{\lambda}_i\widehat{r}\right)^2}\right)\widehat{\lambda}_i = \phi. \tag{10}$$

Eq. (10) can be transformed into Eq. (11),

$$\widehat{\lambda}_i^2 a_i + \widehat{\lambda}_i b_i + c_i = 0, \tag{11}$$

where

$a_i = \widehat{r}^2\phi s_i,$
$b_i = -2\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i\right)\widehat{r}\phi s_i,$
$c_i = -\left(\left(\widehat{r} - \phi s_i\right)\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i\right) + \widetilde{\lambda}_i\widetilde{r}_i^2\right)\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i\right).$

Solving Eq. (11), we can get

$$\begin{aligned}\widehat{\lambda}_i &= \frac{-b_i - 2\sqrt{{b_i}^2 - 4a_i c_i}}{2a_i} \\ &= \frac{t_i}{\widehat{r}} - \frac{1}{\widehat{r}}\sqrt{\frac{t_i\left(\widehat{r}t_i + \widetilde{\lambda}_i\widetilde{r}_i^2\right)}{s_i\phi}},\end{aligned} \tag{12}$$

where $t_i = s_i - \widetilde{\lambda}_i\widetilde{r}_i$. In fact, Eq. (11) has two solutions, we keep the left one [12].

For $v_i \in G_2$, based on Eqs. (5) and (9), we have

$$\frac{\widehat{r}}{s_i} + \frac{\left(2\widehat{\lambda}_i\widehat{r}^2 + \widetilde{\lambda}_i\widetilde{r}_i^2\right)\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i\right) - \widehat{\lambda}_i^2\widehat{r}^3}{\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i\right)\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i - \widehat{\lambda}_i\widehat{r}\right)^2} = \phi. \tag{13}$$

Eq. (13) can be transformed into Eq. (14),

$$\widehat{\lambda}_i^2 a_i + \widehat{\lambda}_i b_i + c_i = 0, \tag{14}$$

where

$$a_i = (\phi - \widehat{r}/s_i)\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i\right)\widehat{r}^2 + \widehat{r}^3,$$
$$b_i = -2\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i\right)\widehat{r}\left((\phi - \widehat{r}/s_i)\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i\right) + \widehat{r}\right),$$
$$c_i = \left(s_i - \widetilde{\lambda}_i\widetilde{r}_i\right)\left((\phi - \widehat{r}/s_i)\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i\right)^2 - \widetilde{\lambda}_i\widetilde{r}_i^2\right).$$

Solving Eq. (14), we get

$$\widehat{\lambda}_i = \frac{t_i}{\widehat{r}} - \frac{1}{\widehat{r}}\sqrt{\frac{t_i\left(\widehat{r}t_i + \widetilde{\lambda}_i\widetilde{r}_i^2\right)}{\phi t_i + \frac{\widehat{r}}{s_i}\widetilde{\lambda}_i\widetilde{r}_i}}, \tag{15}$$

where $t_i = s_i - \widetilde{\lambda}_i\widetilde{r}_i$.

For $v_i \in G_3$, based on Eqs. (6) and (9), we obtain

$$\frac{1}{t_i}\left(\widehat{r} + \frac{\widehat{\lambda}_i\widehat{r}^2 + \widetilde{\lambda}_i\widetilde{r}_i^2}{\left(t_i - \widehat{\lambda}_i\widehat{r}\right)} + \frac{\widehat{\lambda}_i\widehat{r}^2 t_i + \widetilde{\lambda}_i^2\widehat{r}\widetilde{r}_i^2}{\left(t_i - \widehat{\lambda}_i\widehat{r}\right)^2}\right) = \phi, \tag{16}$$

and from Eq. (16) we get

$$\widehat{\lambda}_i^2 a_i + \widehat{\lambda}_i b_i + c_i = 0, \tag{17}$$

where

$$a_i = \phi t_i\widehat{r}^2,$$
$$b_i = \widehat{r}\left(\widehat{r}t_i - \phi {t_i}^2 + \widetilde{\lambda}_i\widetilde{r}_i^2 + \phi t_i\widehat{\lambda}_i\widehat{r}\right),$$
$$c_i = -t_i\left(\widehat{r}t_i - \phi {t_i}^2 + \widetilde{\lambda}_i\widetilde{r}_i^2\right) - \widehat{r}^2 t_i - \widetilde{\lambda}_i\widehat{r}\widetilde{r}_i^2,$$

based on Eq. (17), we can obtain

$$\widehat{\lambda}_i = \frac{t_i}{\widehat{r}} - \frac{1}{\widehat{r}}\sqrt{\frac{\widehat{r}t_i + \widetilde{\lambda}_i\widetilde{r}_i^2}{\phi}}, \tag{18}$$

where $t_i = s_i - \widetilde{\lambda}_i\widetilde{r}_i$.

By deriving $\widehat{\lambda}_i$, we have obtained

$$\widehat{\lambda}_i = \begin{cases} \frac{t_i}{\widehat{r}} - \frac{1}{\widehat{r}}\sqrt{\frac{t_i\left(t_i\widehat{r} + \widetilde{\lambda}_i\widetilde{r}_i^2\right)}{\left(\phi + \tau_i\right)s_i}}, & v_i \in G_1; \\ \frac{t_i}{\widehat{r}} - \frac{1}{\widehat{r}}\sqrt{\frac{t_i\left(\widehat{r}t_i + \widetilde{\lambda}_i\widetilde{r}_i^2\right)}{\left(\phi + \tau_i\right)t_i + \frac{\widehat{r}}{s_i}\widetilde{\lambda}_i\widetilde{r}_i}}, & v_i \in G_2; \\ \frac{t_i}{\widehat{r}} - \frac{1}{\widehat{r}}\sqrt{\frac{t_i\widehat{r} + \widetilde{\lambda}_i\widetilde{r}_i^2}{\phi}}, & v_i \in G_3; \end{cases} \tag{19}$$

where $t_i = s_i - \widetilde{\lambda}_i\widetilde{r}_i$. Next, we will solve s_i for all $1 \le i \le n$.

We take the partial derivative with respect to s_i, that is

$$\frac{\partial\left(\widehat{\lambda}_i T_i\right)}{\partial s_i} = \phi\frac{\partial\varphi}{\partial s_i} + \tau\frac{\partial\psi}{\partial s_i} = -\tau\alpha_i {s_i}^{\alpha_i - 1}. \tag{20}$$

From Eq. (20) we can get

$$\tau = -\frac{\widehat{\lambda}_i}{\alpha_i {s_i}^{\alpha_i - 1}}\frac{\partial T_i}{\partial s_i}, \tag{21}$$

where

$$\frac{\partial T_i}{\partial s_i} = \begin{cases} -\frac{\widehat{r}}{{s_i}^2} - \frac{\left(\widehat{\lambda}_i\widehat{r}^2 + \widetilde{\lambda}_i\widetilde{r}_i^2\right)\left(\widehat{\lambda}_i\widehat{r} + \widetilde{\lambda}_i\widetilde{r}_i\right)}{{s_i}^2\left(t_i - \widehat{\lambda}_i\widehat{r}\right)^2}, & v_i \in G_1; \\ -\frac{\widehat{r}}{{s_i}^2} - \frac{\left(\widehat{\lambda}_i\widehat{r}^2 + \widetilde{\lambda}_i\widetilde{r}_i^2\right)\widehat{\lambda}_i\widehat{r}}{{t_i}^2\left(t_i - \widehat{\lambda}_i\widehat{r}\right)^2}, & v_i \in G_2; \\ -\frac{\widehat{r}}{{t_i}^2} - \frac{\left(\widehat{\lambda}_i\widehat{r}^2 + \widetilde{\lambda}_i\widetilde{r}_i^2\right)\widehat{\lambda}_i\widehat{r}}{{t_i}^2\left(t_i - \widehat{\lambda}_i\widehat{r}\right)^2}, & v_i \in G_3, \end{cases} \tag{22}$$

and $t_i = s_i - \widetilde{\lambda}_i\widetilde{r}_i$.

By deriving $\widehat{\lambda}_i$ and s_i, we have obtained Eqs. (19) and (21). Our target is to solve $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n, s_1, s_2, \ldots, s_n$ based on Eqs. (19) and (21), meanwhile the solved $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n, s_1, s_2, \ldots, s_n$ can satisfy constraints Eqs. (2) and (3). Eqs. (19) and (21) include $2n + 2$ variables, which are $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n, s_1, s_2, \ldots, s_n, \phi$ and τ. Our problem is a multi-constrained and multi-variable problem, and is unlikely to get a closed-form solution. So we have to analyze Eqs. (19) and (21) further.

4.2. Analysis

Since the closed-form solution is difficult to obtain, we could work out the numerical solution. By observing Eqs (19) and (21), we find that $\widehat{\lambda}_i$ could be viewed as a function of ϕ and s_i, and τ could be viewed as a function of $\widehat{\lambda}_i$ and s_i. Taking $\widehat{\lambda}_i$ into the function of τ, then τ is a function of ϕ and s_i. Let

$$\widehat{\lambda}_i = f_i(\phi, s_i), \tag{23}$$

and

$$\tau = g_i(\widehat{\lambda}_i, s_i) = g_i(f_i(\phi, s_i), s_i) = -\frac{\widehat{\lambda}_i\frac{\partial\widehat{T}_i}{\partial s_i}}{\alpha_i {s_i}^{\alpha_i - 1}}. \tag{24}$$

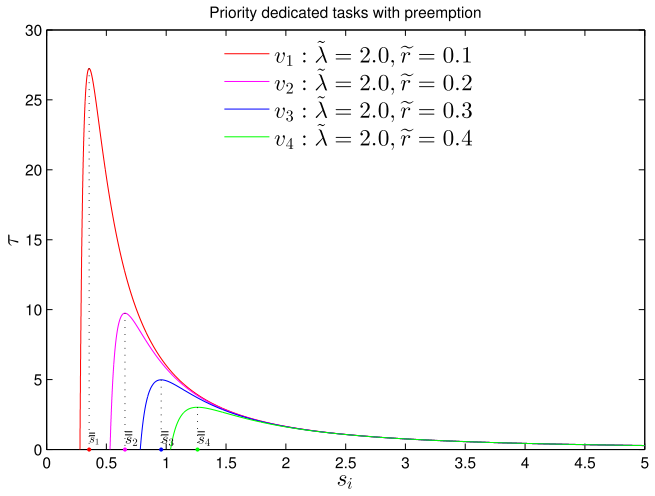

Fig. 2. The function image of $g_i(f_i(\phi, s_i), s_i)$ when $\phi = 7$.

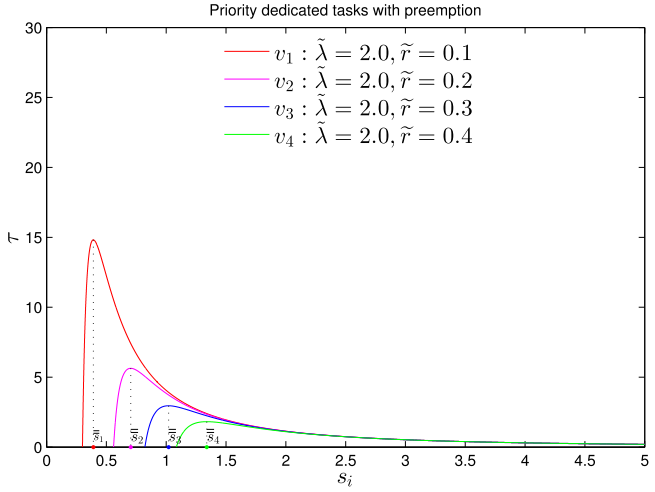

Fig. 3. The function image of $g_i(f_i(\phi, s_i), s_i)$ when $\phi = 5$.

Observing Eq. (24) we can find that if the value of ϕ is fixed, the function $g_i(\widehat{\lambda}_i, s_i)$ includes only one variable s_i. This observation is very helpful. Although s_i for different nodes may be different, τ for all nodes is the same because it is a Lagrange multiplier. If τ could determine s_i, then τ could also determine s_j where $0 \leq i, j \leq n$. In other words, τ may be able to determine s_i for all $0 \leq i \leq n$.

In order to analyze the relationship between τ and s_i, we first need to determine the monotonicity of τ with respect to s_i. $\tau = g_i(\widehat{\lambda}_i, s_i)$ is a high power and nonlinear function of s_i, which is difficult to analyze the monotonicity based on mathematical derivation. We assign $0.0001, 0.0002, \ldots, 5$ to s_i respectively, and then obtain the corresponding $g_i(f_i(\phi, 0.0001), 0.0001), g_i(f_i(\phi, 0.0002), 0.0002), \ldots, g_i(f_i(\phi, 5), 5)$. By drawing the points $(0.0001, g_i(f_i(\phi, 0.0001), 0.0001)), \ldots, (5, g_i(f_i(\phi, 5), 5))$ on the coordinate axis, the monotonicity can be analyzed based on the graph of $g_i(\widehat{\lambda}_i, s_i)$.

Fig. 2 shows graphs $g_i(\widehat{\lambda}_i, s_i)$ of four nodes all belonging to G_3, and each node is preloaded with different amounts of tasks, where ϕ is set to 7. Meanwhile, the nodes shown in Figs. 3 and 4 are the same as that shown in Fig. 2 expect that ϕ is set to 5 and 3, respectively. From Figs. 2 to 4, we summarize a rule and describe it as Rule 1. In fact, in order to verify this rule, many related experiments have been tested, and we can obtain the same result. Owing to limited space, it is impossible to list all of these experiments.

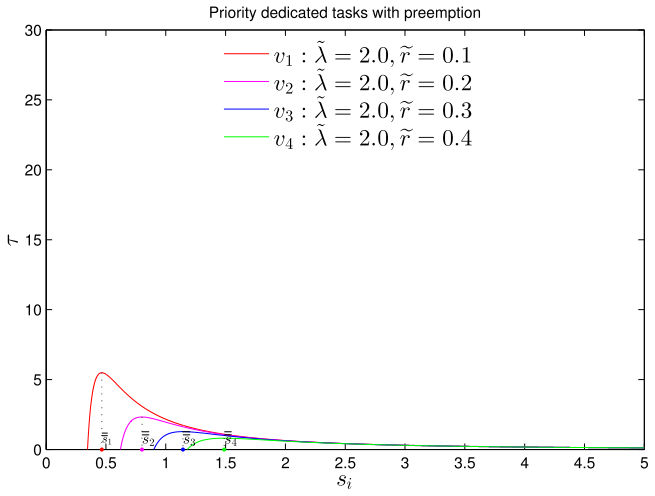

Fig. 4. The function image of $g_i(f_i(\phi, s_i), s_i)$ when $\phi = 3$.

Rule 1: With each fixed value of ϕ, the value of function $\tau = g_i(\widehat{\lambda}_i, s_i)$ initially increased with an increase in s_i and then decreased as s_i continued to increase, for all $1 \leq i \leq n$.

In Figs. 2 to 4, $\bar{\bar{s}}_i$ represents the corresponding value of s_i when $g_i(\widehat{\lambda}_i, s_i)$ is maximum, namely $g_i(\widehat{\lambda}_i, s_i) \leq g_i(\widehat{\lambda}_i, \bar{\bar{s}}_i)$. Obviously, each $g_i(\widehat{\lambda}_i, s_i)$ is a convex function when s_i is located at $[\bar{\bar{s}}_i, +\infty]$. On the basis of convex optimization theory, $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n, s_1, s_2, \ldots, s_n$ will be the optimal solution, if $\sum_{i=1}^{n} f_i(\phi, s_i) = \widehat{\lambda}, \overline{P} = \sum_{i=1}^{n} (s_i^{\alpha_i} + P_i^*)$ and $s_i \in [\bar{\bar{s}}_i, +\infty]$ for all $1 \leq i \leq n$ could be met simultaneously. To obtain the optimal solution, two steps are designed. First, given a value of ϕ, find a τ that can be used to determine the speeds $s_1, s_2, \ldots, s_n$ under the power constraint Eq. (3). Then, based on the first step, by adjusting the value of ϕ, find the appropriate ϕ and τ that simultaneously meet the constraints equation (3) and equation (2). The following two paragraphs describe the two steps.

This paragraph explains step one. For $g_i(\widehat{\lambda}_i, s_i)$, $(\bar{\bar{s}}_i, g_i(\widehat{\lambda}_i, \bar{\bar{s}}_i))$ is a turning point, which makes $g_i(\widehat{\lambda}_i, s_i)$ obtain the maximum value, i.e., $g_i(\widehat{\lambda}_i, s_i) \leq g_i(\widehat{\lambda}_i, \bar{\bar{s}}_i)$. The Lagrange multiplier τ should have the same value for all $g_i(\widehat{\lambda}_i, s_i)$. Thus, the upper bound of τ is the smallest maximum value of all $g_i(\widehat{\lambda}_i, s_i)$, namely $\tau \leq \text{Min}\left\{g_1(\widehat{\lambda}_1, \bar{\bar{s}}_1), g_2(\widehat{\lambda}_2, \bar{\bar{s}}_2), \ldots, g_n(\widehat{\lambda}_n, \bar{\bar{s}}_n)\right\}$. At the right side of $\bar{\bar{s}}_i$, $g_i(\widehat{\lambda}_i, s_i)$ is monotonically decreasing. In other words, with a given τ, the corresponding s_i can be immediately obtained by using binary search. Power P_i is a monotonically increasing function of s_i for all $1 \leq i \leq n$. Since τ could determine $s_1, s_2, \ldots, s_n$ $(s_i \in [\bar{\bar{s}}_i, +\infty])$, $\sum_{i=1}^{n} P_i = \overline{P}$ can be obtained by adjusting τ.

At the first step we can obtain a value of τ which make $s_1, s_2, \ldots, s_n$ satisfy the power constraint Eq. (3). The remaining work is to find the appropriate ϕ that satisfies $\sum_{i=1}^{n} f_i(\phi, s_i) = \widehat{\lambda}$. $\widehat{\lambda}_i = f_i(s_i, \phi)$ contains two independent variables, s_i and ϕ. Given a ϕ, based on step one, we can find $s_1, s_2, \ldots, s_n$ that satisfy power constraint Eq. (3). If ϕ is changed, we can still obtain all new s_i satisfying Eq. (3). Of course, for different ϕ, the obtained $s_1, s_2, \ldots, s_n$ are different. It seems difficult that finding the appropriate ϕ and τ satisfy Eqs. (2) and (3) at the same time. However, by examining a large number of experimental data, we observed that as long as all s_i locate at $[\bar{\bar{s}}_i, +\infty]$ and $s_1, s_2, \ldots, s_n$

meet Eq. (3), the rangeability of s_i remains relatively stable even if changing ϕ will change the value of s_i. This finding suggests that the value of ϕ is the dominant factor affecting the value of $\widehat{\lambda}_i$. On this basis, we introduce the second observed rule.

Rule 2: On the premise of all s_i locating at $[\bar{\bar{s}}_i, +\infty]$ and $s_1, s_2, \ldots, s_n$ satisfying Eq. (3), $\widehat{\lambda}_i$ will increase monotonically with ϕ, indicating that $\sum_{i=1}^{n} \widehat{\lambda}_i$ will increase monotonically with ϕ.

According to above analyses and Rules 1 and 2, we can design algorithms to find the optimal solution $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n, s_1, s_2, \ldots, s_n$, and corresponding Lagrange multipliers ϕ and τ. These algorithms will be introduced in next section.

4.3. Algorithm design

In this section, we will design algorithms to find the optimal solution $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n, s_1, s_2, \ldots, s_n$, and the corresponding Lagrange multipliers ϕ and τ. Based on Rules 1 and 2 mentioned in Section 4.2, our method is divided to four steps for solving problem, which are described as follows.

1. Solve the searching region for s_i.
2. Find the turning point $\bar{\bar{s}}_i$ on the searching region of s_i.
3. Fix the value of ϕ, and find a τ on region $[\bar{\bar{s}}, +\infty]$ which can make $s_1, s_2, \ldots, s_n$ satisfy Eq. (3).
4. Based on steps 1, 2 and 3, adjust ϕ until both Eqs. (2) and (3) are satisfied.

The searching region for s_i is determined by $\widehat{\lambda}_i = f_i(\phi, s_i) > 0$, and could be solved based on Eq. (19).

For $v_i \in G_1$, we can get

$$\frac{s_i - \widetilde{\lambda}_i\widetilde{r}_i}{\widehat{r}} > \frac{1}{\widehat{r}}\sqrt{\frac{\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i\right)\left(\widehat{r}\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i\right) + \widetilde{\lambda}_i\widetilde{r}_i^2\right)}{s_i\phi}}.$$

Then, we will have

$$a_i{s_i}^2 + b_i s_i + c_i \geq 0,$$

where

$$a_i = \phi,\ b_i = -\left(\widetilde{\lambda}_i\widetilde{r}_i\phi + \widehat{r}\right),\ c_i = \widetilde{r}_i\left(\widehat{r} - \widetilde{r}_i\right).$$

We set that

$$\Delta_i = {b_i}^2 - 4a_ic_i = \left(\widetilde{\lambda}_i\widetilde{r}_i\phi + \widehat{r}\right)^2 - 4\phi\widetilde{r}_i\left(\widehat{r} - \widetilde{r}_i\right).$$

As we know, in real situations of distributed environments, we always expect that a node whose CPU speed is improved could be assigned with more tasks. We called it as "more energy more tasks", which indicates that $\widehat{\lambda}_i$ increases with the increase of s_i. Based on "more energy more tasks", we define the searching region for s_i ($v_i \in G_1$) as follows:

$$s_i \geq \begin{cases} \dfrac{\left(\widetilde{\lambda}_i\widetilde{r}_i\phi + \widehat{r}\right)}{2\phi}, \Delta_i \leq 0; \\ \dfrac{1}{2\phi}\left(\sqrt{\left(\widetilde{\lambda}_i\widetilde{r}_i\phi + \widehat{r}\right)^2 - 4\phi\widetilde{\lambda}_i\widetilde{r}_i\left(\widehat{r} - \widetilde{r}_i\right)}\right. \\ \qquad \left. + \widetilde{\lambda}_i\widetilde{r}_i\phi + \widehat{r}\right), \Delta_i > 0. \end{cases} \tag{25}$$

Similarly, for $v_i \in G_2$, we can obtain

$$a_i{s_i}^3 + b_i{s_i}^2 + c_i s_i + d_i \geq 0, \tag{26}$$

where

$$a_i = \phi,\ b_i = -\left(2\phi\widetilde{\lambda}_i\widetilde{r}_i + \widehat{r}\right),$$

$$c_i = \widetilde{\lambda}_i\widetilde{r}_i\left(2\widehat{r} + \lambda\phi\widetilde{\lambda}_i\widetilde{r}_i - \widetilde{r}_i\right),$$

$$d_i = -\left(\widetilde{\lambda}_i\widetilde{r}_i\right)^2\widehat{r}.$$

Eq. (26) is a cubic polynomial equation about s_i. To solve it we need to use Fan's formula [24]. Let

$$A_i = {b_i}^2 - 3a_ic_i,\ B_i = b_ic_i - 9a_id_i,\ C_i = {c_i}^2 - 3b_id_i.$$

Let $\Delta_i = {B_i}^2 - 4A_iC_i$ as the discriminant. Taking the value of Δ_i and the "more energy more tasks" into consideration, we obtain the searching region for s_i ($v_i \in G_2$) as follows:

$$s_i \geq \begin{cases} \dfrac{-b_i - \left(\sqrt[3]{Y_1} + \sqrt[3]{Y_2}\right)}{3a_i}, \Delta_i > 0; \\ \dfrac{-b_i}{3a_i} + \dfrac{B_i}{A_i}, \Delta_i = 0; \\ \dfrac{-b_i + \sqrt{A}\left(\cos\frac{\theta}{3} + \sqrt{3}\sin\frac{\theta}{3}\right)}{3a}, \Delta_i < 0, \end{cases} \tag{27}$$

where

$$Y_{1,2} = A_ib_i + 3a_i\left(\frac{-B_i \pm \sqrt{{B_i}^2 - 4A_iC_i}}{2}\right).$$

For $v_i \in G_3$, we obtain

$$\phi\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i\right)^2 - \widehat{r}\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i\right) - \widetilde{\lambda}_i\widetilde{r}_i^2 \geq 0. \tag{28}$$

From Eq. (28) we obtain the searching region for s_i ($v_i \in G_3$), which is

$$s_i \geq \frac{\widehat{r} + \sqrt{\widehat{r}^2 + 4\phi\widetilde{\lambda}_i\widetilde{r}_i^2}}{2\phi} + \widetilde{\lambda}_i\widetilde{r}_i, \tag{29}$$

We have solved the searching region for s_i ($0 \leq i \leq n$). The next step is to find the turning point $\bar{\bar{s}}_i$ on the searching region of s_i.

From Figs. 2 to 4, we observer that $g_i(\widehat{\lambda}_i, s_i)$ is an increasing function when s_i locates at the left side of $\bar{\bar{s}}_i$, and $g_i(\widehat{\lambda}_i, s_i)$ is a decreasing function when s_i locates at the right side of $\bar{\bar{s}}_i$. This means that the derivative of $g_i(\widehat{\lambda}_i, s_i)$ is larger than 0, $g_i{}'\left(\widehat{\lambda}_i, s_i\right) > 0$, if $s_i < \bar{\bar{s}}_i$, and $g_i{}'\left(\widehat{\lambda}_i, s_i\right) < 0$ if $s_i > \bar{\bar{s}}_i$. Therefore, $\bar{\bar{s}}_i$ can be solved using binary search based on $g_i{}'\left(\widehat{\lambda}_i, s_i\right)$. Due to the attention only on s_i, for the sake of simplicity, we replace the partial derivative of $f_i(\phi, s_i)$ with $f_i{}'(s_i)$, and replace the partial derivative of $g_i(\widehat{\lambda}, s_i)$ with $g_i{}'(\widehat{\lambda}, s_i)$.

$$g'_i\left(\widehat{\lambda}_i, s_i\right) = \frac{\dfrac{\partial^2\widehat{T}_i}{\partial {s_i}^2}f_i(s_i) - \left(\dfrac{(\alpha_i - 1)f_i(s_i)}{s_i} - f_i'(s_i)\right)\dfrac{\partial\widehat{T}_i}{\partial s_i}}{\alpha_i {s_i}^{\alpha_i - 1}} \tag{30}$$

where

$$f_i'(s_i) = \begin{cases} \dfrac{1}{\widehat{r}} - \dfrac{\left(\widehat{r}t_i + d_i\right)s_i - t_id_i}{2\widehat{r}\sqrt{\phi t_id_i{s_i}^3}}, v_i \in G_1; \\ \dfrac{1}{\widehat{r}} - \dfrac{\left(t_is_i\widehat{r} + d_i\left(t_i + s_i\right)\right)\widetilde{\lambda}_i\widetilde{r}_i\widehat{r}}{\left(\phi t_is_i + \widetilde{\lambda}_i\widetilde{r}_i\widehat{r}\right)^{3/2}2\sqrt{d_it_is_i}}, v_i \in G_2; \\ \dfrac{1}{\widehat{r}} - \dfrac{1}{2\sqrt{d_i\phi}}, v_i \in G_3; \end{cases}$$

$d_i = \widehat{r}t_i + \widetilde{\lambda}_i\widetilde{r}_i^2$, $t_i = s_i - \widetilde{\lambda}\widetilde{r}_i$. For $v_i \in G_1$, we have

$$-\frac{\partial^2 \widehat{T}_i}{\partial {s_i}^2} = \frac{-2\widehat{r}}{{t_i}^3} + \frac{2t_i - \widehat{\lambda}_i\widehat{r}}{{t_i}^2\left(t_i - \widehat{\lambda}_i\widehat{r}\right)^2} \times \left(\left(\widehat{\lambda}_i\widehat{r}^2 - \left(\widehat{\lambda}_i\widehat{r}^2 + \widetilde{\lambda}_i\widetilde{r}_i^2\right) \times \left(\frac{1}{t_i} + \frac{1 - \widehat{\lambda}_i\widehat{r}}{t_i - \widehat{\lambda}_i\widehat{r}}\right)\right) - \frac{1}{t_i}\frac{\widehat{\lambda}_i\widehat{r}^2 + \widetilde{\lambda}_i\widetilde{r}_i^2}{\left(t_i - \widehat{\lambda}_i\widehat{r}\right)} \times \left(\frac{1}{{t_i}^2} + \frac{1 - \widehat{\lambda}_i\widehat{r}}{\left(t_i - \widehat{\lambda}_i\widehat{r}\right)^2}\right).$$

For $v_i \in G_2$, we have

$$-\frac{\partial^2 \widehat{T}_i}{\partial {s_i}^2} = \frac{\widehat{r}}{\lambda {s_i}^2}\left(f_i'(s_i) - \frac{2}{s_i}\widehat{\lambda}_i\right) + \frac{H_i}{\lambda t_i\left(t_i - \widehat{\lambda}_i\widehat{r}\right)^2},$$

where

$$H_i = f_i'(s_i)\left(2 - \frac{\widehat{\lambda}_i\widehat{r}}{t_i}\right) \times \left(2\widehat{\lambda}_i\widehat{r}^2 + \widetilde{\lambda}_i\widetilde{r}_i^2\right) - \frac{\widehat{\lambda}_i\left(\widehat{\lambda}_i\widehat{r}^2 + \widetilde{\lambda}_i\widetilde{r}_i^2\right)}{t_i\left(t_i - \widehat{\lambda}_i\widehat{r}\right)} \times \left(6t_i + \widehat{\lambda}_i\widehat{r}\left(\frac{2\widehat{\lambda}_i\widehat{r}}{t_i} - 6\right) + \widehat{r}^2 f_i'(s_i)\left(\widehat{\lambda}_i\widehat{r} - 3t_i\right)\right).$$

For $v_i \in G_3$, we have

$$-\frac{\partial^2 \widehat{T}_i}{\partial {s_i}^2} = \frac{-2\widehat{r}}{{t_i}^3} + \frac{2t_i - \widehat{\lambda}_i\widehat{r}}{{t_i}^2\left(t_i - \widehat{\lambda}_i\widehat{r}\right)^2} \times \left(f_i'(s_i)\widehat{r}^2 - \left(\widehat{\lambda}_i\widehat{r}^2 + \widetilde{\lambda}_i\widetilde{r}_i^2\right)\left(\frac{1}{t_i} + \frac{1 - f_i'(s_i)\widehat{r}}{t_i - \widehat{\lambda}_i\widehat{r}}\right)\right) - \left(\widehat{\lambda}_i\widehat{r}^2 + \widetilde{\lambda}_i\widetilde{r}_i^2\right) \times \left(\frac{\left(1 - f_i'(s_i)\widehat{r}\right){t_i}^2 + \left(t_i - \widehat{\lambda}_i\widehat{r}\right)^2}{{t_i}^3\left(t_i - \widehat{\lambda}_i\widehat{r}\right)^3}\right).$$

The binary search will be used many times. In order to avoid repeatedly using a list of search methods, we define it in Algorithm 1. According to this definition, $\bar{\bar{s}}_i$ could be solved by biSearch($lows_i$, $maxs_i$, $g_i'\left(\widehat{\lambda}_i, s_i\right) > 0$), where $lows_i$ is supported by Eqs. (25), (27), and (29); $maxs_i$ can be an appropriate value, for instance, it could be set to 5 because at present most processor's maximal frequency is less than 5 GHz.

Algorithm 1 biSearch(lb, ub, $criterion$)

Require: lb, ub, $criterion$.
Ensure: var.
1: **while** $(ub - lb > \varepsilon)$ **do**
2: $var \leftarrow (ub + lb)/2$;
3: **if** $criterion$ **then**
4: $lb \leftarrow var$;
5: **else**
6: $ub \leftarrow var$;
7: **end if**
8: **end while**
9: **return** var.

In this paragraph, we will introduce how to find a τ on region $[\bar{\bar{s}}, +\infty]$ which can make $s_1, s_2, \ldots, s_n$ satisfy Eq. (3). τ is a common Lagrange multiplier. Since τ is considered as $\tau = g_i(\widehat{\lambda}_i, s_i)$, the upper bound of τ is the smallest maximum value of all $g_i(\widehat{\lambda}_i, s_i)$ (Algorithm 2). On region $[\bar{\bar{s}}, +\infty]$, $g_i(\widehat{\lambda}_i, s_i)$ is a decreasing function of s_i. In other words, given a τ, the corresponding s_i can be solved by binary search, i.e., biSearch($\bar{\bar{s}}_i$, $maxs_i$, $g_i(\widehat{\lambda}_i, s_i) > \tau$). Since τ is a monotonous function of s_i, τ is also a monotonous function of $\sum_{i=1}^{n}(s_i^{\alpha_i} + P_i^*)$. Algorithm 3 shows how to find a τ which can make $s_1, s_2, \ldots, s_n$ satisfy $\sum_{i=1}^{n}(s_i^{\alpha_i} + P_i^*) \leq \overline{P}$. In Algorithm 3, the s_i shown in line 7 is calculated by biSearch($\bar{\bar{s}}_i$, $maxs_i$, $g_i(\widehat{\lambda}_i, s_i) > \tau$).

Algorithm 2 findUBof_$\tau(\phi, \bar{\bar{s}}_1, \ldots, \bar{\bar{s}}_n)$

Require: $\phi, \bar{\bar{s}}_1, \ldots, \bar{\bar{s}}_n$.
Ensure: $uBof\_\tau$.
1: $uBof\_\tau \leftarrow \gamma$;
2: **for** $(1 \leftarrow i; i \leq n; i++)$ **do**
3: $temp \leftarrow g_i(f_i(\bar{\bar{s}}_i), \bar{\bar{s}}_i)$;
4: **if** $temp < \gamma$ **then**
5: $uBof\_\tau \leftarrow temp$;
6: **end if**
7: **end for**
8: **return** $uBof\_\tau$.

Algorithm 3 calculateAlls$_i$

Require: $\phi, \overline{P}$.
Ensure: $s_1, s_2, \ldots, s_n$.
1: $\bar{\bar{s}}_1, \bar{\bar{s}}_2, \ldots, \bar{\bar{s}}_n$;
2: **for** $(1 \leftarrow i; i \leq n; i \leftarrow i+1)$ **do**
3: $\bar{\bar{s}}_i \leftarrow$ biSearch($lows_i$, $maxs_i$, $g_i'\left(\widehat{\lambda}_i, s_i\right) > 0$));
4: **end for**
5: $ub \leftarrow$ findUBof_$\tau(\phi, \bar{\bar{s}}_1, \bar{\bar{s}}_2, \ldots, \bar{\bar{s}}_n)$
6: $lb \leftarrow 0$;
7: $\tau \leftarrow$biSearch(lb, ub, $\sum_{i=1}^{n}(s_i^{\alpha_i} + P_i^*) < \overline{P}$);
8: **return** $\tau, s_1, s_2, \ldots, s_n$.

Based on the Rule 2 mentioned in Section 4.2, we can use binary search to obtain the appropriate ϕ, which can then be used to calculate $\tau, s_1, s_2, \ldots, s_n$ and $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n$ that meet Eqs. (2) and (3) at the same time.

Algorithm 4 calculateLBof_ϕ

Require: $\widetilde{\lambda}_1, \ldots, \widetilde{\lambda}_n, s_1, \ldots, s_n, \widetilde{r}_1, \widetilde{r}_2, \ldots, \widetilde{r}_n, \overline{P}$.
Ensure: $lBof\_\phi$.
1: $uBof\_\phi \leftarrow 1.0$;
2: **repeat**
3: $\phi \leftarrow 2\phi$;
4: **for** $(1 \leftarrow i; i \leq n; i \leftarrow i+1)$ **do**
5: $\bar{\bar{s}}_i \leftarrow$ biSearch($lows_i$, $maxs_i$, $g_i'\left(\widehat{\lambda}_i, s_i\right) > 0$));
6: **end for**
7: $\tau \leftarrow$ findUBof_$\tau(\phi, \bar{\bar{s}}_1, \bar{\bar{s}}_2, \ldots, \bar{\bar{s}}_n)$;
8: **for** $(1 \leftarrow i; i \leq n; i++)$ **do**
9: $s_i \leftarrow$ biSearch($\bar{\bar{s}}_i$, $maxs_i$, $g_i(\widehat{\lambda}_i, s_i) > \tau$);
10: **end for**
11: **until** $(\sum_{i=1}^{n}(s_i^{\alpha_i} + P_i^*) \geq \overline{P})$
12: $lb \leftarrow 0, ub \leftarrow \phi$;
13: $\phi \leftarrow$biSearch(lb, ub, $\sum_{i=1}^{n}(s_i^{\alpha_i} + P_i^*) \geq \overline{P}$);
14: //The s_i in line 13 is calculated by
//$\bar{\bar{s}}_i \leftarrow$ biSearch($lows_i$, $maxs_i$, $g_i'\left(\widehat{\lambda}_i, s_i\right) > 0$));
//$\tau \leftarrow$ findUBof_$\tau(\phi, \bar{\bar{s}}_1, \bar{\bar{s}}_2, \ldots, \bar{\bar{s}}_n)$;
//$s_i \leftarrow$ biSearch($\bar{\bar{s}}_i$, $maxs_i$, $g_i(\widehat{\lambda}_i, s_i) > \tau$);
15: **return** $s_1, \ldots, s_n, \phi_B, \widehat{\lambda}_B \leftarrow \sum_{i=1}^{n} f_i(s_i, \phi)$.

As mentioned, the major steps of our method can be summed up as: given a ϕ, search a τ that can be used to find $s_1, s_2, \ldots, s_n$ satisfying Eq. (3); and then adjust ϕ until both Eqs. (2) and (3) are satisfied. We know that $\sum_{i=1}^{n} f_i(\phi, s_i)$ decreases monotonically when ϕ decreases, therefore a small value of $\widehat{\lambda}$ will be matched with a small ϕ. However, Figs. 2 to 4 suggest that when ϕ is small, there may not exist a τ that makes all s_i not only locate at the right side of $\overline{\overline{s_i}}$ but also satisfy Eq. (3), i.e., $\overline{P} = \sum_{i=1}^{n} (s_i^{\alpha_i} + P_i^*)$. For instance, from Figs. 2 to 4, $g_1(f_1(\phi, \overline{\overline{s}}_1), \overline{\overline{s}}_1)$ is about 28, 15 and 5 when ϕ is 7, 5 and 3, respectively. If ϕ becomes smaller, then τ will also become smaller, causing s_i to become larger. If ϕ is too small, then $\sum_{i=1}^{n} (s_i^{\alpha_i} + P_i^*) \leq \overline{P}$ cannot be satisfied.

To deal with the situation with a small $\widehat{\lambda}$, we will find a threshold ϕ_B. When $\phi \geq \phi_B$, there exists a τ that makes $s_1, s_2, \ldots, s_n$ not only locate at the right side of $\overline{\overline{s_i}}$ but also satisfy Eq. (3). When $\phi < \phi_B$, such a τ does not exist. According to whether τ exists, ϕ_B can be solved by binary search. Assume that we get $\sum_{i=1}^{n} f_i(\phi, s_i) = \widehat{\lambda}_B$ and $s_1, s_2, \ldots, s_n = s_{1B}, s_{2B}, \ldots, s_{nB}$ satisfying $\sum_{i=1}^{n} (s_i^{\alpha_i} + P_i^*) = \overline{P}$, when $\phi = \phi_B$. When $\widehat{\lambda} < \widehat{\lambda}_B$, we still adopt $s_{1B}, s_{2B}, \ldots, s_{nB}$, that is, keep the speeds of nodes unchanged. Based on $s_{1B}, s_{2B}, \ldots, s_{nB}$ and $\sum_{i=1}^{n} \widehat{\lambda}_i = \widehat{\lambda}$, the load balancing $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n$ can be solved by binary search.

Notice that the ϕ_B and $\widehat{\lambda}_B$ are independent of $\widehat{\lambda}$. For a system under analysis, we first calculate the threshold ϕ_B and $\widehat{\lambda}_B$. Algorithm 4 shows how ϕ_B can be obtained. Algorithm 5 shows how the final $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n, s_1, s_2, \ldots, s_n, \phi$ and τ can be determined.

Algorithm 5 caculate_T

Require: $\widetilde{\lambda}_1, \ldots, \widetilde{\lambda}_n, s_1, \ldots, s_n, \widetilde{r}_1, \ldots, \widetilde{r}_n, \widehat{r}, \overline{P}, \widehat{\lambda}$.
Ensure: $\widehat{\lambda}_1, \ldots, \widehat{\lambda}_n, s_1, \ldots, s_n, \phi, \tau$.
1: $s_i, \phi_B, \widehat{\lambda}_B \leftarrow$ *calculateLBof_*ϕ;
2: **if** $(\widehat{\lambda} > \widehat{\lambda}_B)$ **then**
3: $lb \leftarrow \phi_B$;
4: **repeat**
5: $\phi \leftarrow 2\phi$;
6: $s_1, s_2, \ldots, s_n \leftarrow$ *calculateAlls*$_i$;
7: **until** $\widehat{\lambda}_1 + \widehat{\lambda}_2 + \cdots + \widehat{\lambda}_n > \widehat{\lambda}$
8: $ub \leftarrow \phi$;
9: $\phi \leftarrow$biSearch$(lb, ub, \sum_{i=1}^{n} f_i(\phi, s_i) < \widehat{\lambda})$; //The speeds
// $s_1, s_2, \ldots, s_n$ in line 9 are calculated by Algorithm 3.
10: **else**
11: $lb \leftarrow 0, ub \leftarrow \phi_B$;
12: $\phi \leftarrow$biSearch$(lb, ub, \sum_{i=1}^{n} \widehat{\lambda}_i < \widehat{\lambda})$; //Note that speed s_i is not
// considered here.
13: **end if**
14: **return** $\widehat{\lambda}_1, \ldots, \widehat{\lambda}_n, s_1, \ldots, s_n, \phi, \tau, T_1, T_2, \ldots, T_n, T$.

5. Numerical examples

In this section, we demonstrate a number of numerical examples. All parameters in our examples are for illustration only. They can be changed to any others real values.

Example 1. In this example, we consider the situation in which the preloaded tasks of each node are heterogeneous, whereas the priority discipline and dynamic power consumption of each node are homogeneous. We consider a group of $n = 7$ embedded nodes $v_1, v_2, \ldots, v_7$. The base power consumption is $P_i^* = 0.1$ W for all $1 \leq i \leq n$. The preloaded dedicated tasks and α_i of each node are shown in Table 2. The unit of task execution requirement is giga instructions, and the unit of task arrival rate is number of tasks per second. The general tasks execution requirement is $\widehat{r} = 0.35$ (giga instructions), and its arrival rate is $\widehat{\lambda} = 16.5$ per/second. We assume $\overline{P} = 30$ W. In Tables 3, 4, and 5, we show the optimal load distribution $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_7$, the optimal node speed $s_1, s_2, \ldots, s_7$, the node utilizations $\rho_1, \rho_2, \ldots, \rho_7$ and the average general task response time $T_1, T_2, \ldots, T_7$, respectively, for Discipline 1, Discipline 2 and Discipline 3. The overall average task response time of the n nodes is $T = 0.7717698$ s, $T = 0.9770131$ s and $T = 1.0462909$ s, respectively.

Table 2
The input parameters in Example 1.

i	1	2	3	4	5	6	7
$\widetilde{\lambda}_i$	1.0	1.0	2.0	2.0	2.0	2.0	2.0
$\widetilde{r}_i$	0.1	0.15	0.2	0.25	0.3	0.35	0.35
α_i	2.7	2.7	2.7	2.7	2.7	2.7	2.7

Table 3
Numerical data in Example 1 when system priority strategy is Discipline 1.

i	$\widehat{\lambda}_i$	s_i	P_i	ρ	T_i
1	3.3843571	1.7802273	4.8455195	0.7215510	0.6777390
2	3.2485873	1.7719359	4.7860795	0.7263273	0.6868395
3	2.5641919	1.7236169	4.4490062	0.7527584	0.7396214
4	2.2602470	1.7003776	4.2924948	0.7592939	0.7832926
5	1.9304618	1.6686682	4.0847307	0.7644788	0.8448245
6	1.5560704	1.6187616	3.7710801	0.7688745	0.9354861
7	1.5560704	1.6187616	3.7710801	0.7688745	0.9354861

Table 4
Numerical data in Example 1 when system priority strategy is Discipline 2.

m_i	$\widehat{\lambda}_i$	s_i	P_i	ρ	T_i
1	3.5044902	1.7645399	4.7334562	0.7517945	0.8009452
2	3.3534759	1.7585929	4.6914135	0.7527134	0.8184316
3	2.5885737	1.7217729	4.4364556	0.7585209	0.9258621
4	2.2529669	1.7042148	4.3180886	0.7560892	1.0064470
5	1.8871207	1.6782739	4.1469674	0.7510646	1.1212838
6	1.4566834	1.6294367	3.8368121	0.7424892	1.3005808
7	1.4566834	1.6294367	3.8368121	0.7424892	1.3005808

Table 5
Numerical data in Example 1 when system priority strategy is Discipline 3.

m_i	$\widehat{\lambda}_i$	s_i	P_i	ρ	T_i
1	3.5086541	1.7601689	4.7025312	0.7544894	0.8238614
2	3.3605310	1.7559519	4.6728197	0.7552517	0.8469965
3	2.6052192	1.7265133	4.4687668	0.7598126	0.9894422
4	2.2680335	1.7101187	4.3576597	0.7565625	1.0888493
5	1.8932808	1.6829125	4.1772389	0.7502756	1.2283187
6	1.4321355	1.6251766	3.8104921	0.7391488	1.4502760
7	1.4321355	1.6251766	3.8104921	0.7391488	1.4502760

Example 2. In this example, we consider the situation in which the preloaded tasks and the priority discipline of each node are homogeneous, whereas the dynamic power consumption of each node are heterogeneous. We also consider a group of $n = 7$ embedded nodes. The base power consumption is $P_i^* = 0.1$ W for all $1 \leq i \leq n$. The preloaded dedicated tasks and α_i of each node are shown in Table 6. The arrival rate of general task is $\widehat{\lambda} = 11$ per second and its tasks execution requirement is $\widehat{r} = 0.35$ (giga instructions), and $\overline{P} = 40$ W. The significance of Tables 7, 8, 9 is the same as that of Tables 3, 4, 5. The overall average task response time of the n nodes is $T = 0.6543409$ s (Discipline 1), $T = 0.9065909$ s (Discipline 2) and $T = 1.0261497$ s (Discipline 3).

Example 3. In this example, we consider the situation in which the preloaded tasks and dynamic power consumption are homogeneous, whereas the priority disciplines are heterogeneous. We consider a group of $n = 6$ embedded nodes $v_1, v_2, \ldots, v_6$.

Table 6
The input parameters in Example 2.

i	1	2	3	4	5	6	7
$\widetilde{\lambda}_i$	2.0	2.0	2.0	2.0	2.0	2.0	2.0
$\widetilde{r}_i$	0.3	0.3	0.3	0.3	0.3	0.3	0.3
α_i	2.7	2.75	2.8	2.85	2.9	2.95	3.0

Table 7
Numerical data in Example 2 when system priority strategy is Discipline 1.

i	$\widehat{\lambda}_i$	s_i	P_i	ρ	T_i
1	2.0112581	1.8567479	5.4165872	0.7022710	0.6039040
2	1.8397337	1.7787064	4.9728921	0.6993322	0.6229132
3	1.6853880	1.7078066	4.5753640	0.6967333	0.6418618
4	1.5457792	1.6430560	4.2172695	0.6944514	0.6607954
5	1.4188454	1.5836093	3.8929697	0.6924661	0.6797675
6	1.3028176	1.5287329	3.5976655	0.6907591	0.6988433
7	1.1961643	1.4777832	3.3272469	0.6893145	0.7181025

Table 8
Numerical data in Example 2 when system priority strategy is Discipline 2.

i	$\widehat{\lambda}_i$	s_i	P_i	ρ	T_i
1	2.0147694	1.8430858	5.3116237	0.7081435	0.8281885
2	1.8445937	1.7712306	4.9167774	0.7032442	0.8570339
3	1.6899333	1.7055341	4.5587093	0.6985944	0.8862125
4	1.5485460	1.6451083	4.2319435	0.6941737	0.9158489
5	1.4184926	1.5891786	3.9317825	0.6899617	0.9460990
6	1.2980502	1.5370483	3.6540879	0.6859365	0.9771668
7	1.1856296	1.4880648	3.3950767	0.6820740	1.0093296

Table 9
Numerical data in Example 2 when system priority strategy is Discipline 3.

i	$\widehat{\lambda}_i$	s_i	P_i	ρ	T_i
1	2.0359684	1.8471058	5.3423719	0.7106192	0.9248241
2	1.8612456	1.7748791	4.9441116	0.7050823	0.9613408
3	1.7007907	1.7083003	4.5789877	0.6996876	0.9988055
4	1.5522012	1.6464157	4.2413086	0.6943996	1.0375050
5	1.4132456	1.5883294	3.9258478	0.6891743	1.0778459
6	1.2816665	1.5331226	3.6273766	0.6839526	1.1204471
7	1.1548776	1.4797246	3.3399830	0.6786446	1.1663282

Table 10
The input parameters in Example 3.

i	1	2	3	4	5	6
$\widetilde{\lambda}_i$	2.0	2.0	2.0	2.0	2.0	2.0
$\widetilde{r}_i$	0.3	0.3	0.3	0.3	0.3	0.3
α_i	2.8	2.8	2.8	2.8	2.8	2.8
Type	D1	D1	D2	D2	D3	D3

We set that the priority discipline of node v_1, v_2 is Discipline 1 (abbreviated as D1), the priority discipline of nodes v_3, v_4 is Discipline 2 (abbreviated as D2), and the priority discipline of nodes v_5, v_6 is Discipline 3 (abbreviated as D3). We set $P_i^* = 0.1$ W for all $1 \le i \le n$. The preloaded dedicated tasks and α_i of each node are showed in Table 10. The tasks execution requirement is $\widehat{r} = 0.35$ (giga instructions). We assume that we are given $\overline{P} = 35$ W. For comparison, we set the arrival rate of general task is $\widehat{\lambda} = 8$ and $\widehat{\lambda} = 15$ per second, respectively. Tables 11 and 12 show the optimal load distribution, node speed, node utilization, and average general task response time of each node respectively for $\widehat{\lambda} = 8$ and $\widehat{\lambda} = 15$ per second. The overall average task response time of the n nodes is $T = 0.7279155$ s and $T = 2.5403782$ s, respectively.

Example 4. In this example, we consider the situation in which all the preloaded tasks, priority discipline, and dynamic power consumption of each node in the system are heterogeneous. We consider a group of $n = 9$ embedded nodes $v_1, v_2, \ldots, v_9$,

Table 11
Numerical data in Example 3 when $\widehat{\lambda} = 8$ per second.

i	$\widehat{\lambda}_i$	s_i	P_i	ρ	T_i
1	1.7182053	1.7239624	4.6949192	0.6968665	0.6364412
2	1.7182053	1.7239624	4.6949192	0.6968665	0.6364412
3	1.2645688	1.6483945	4.1529501	0.6324936	0.7396511
4	1.2645688	1.6483945	4.1529501	0.6324936	0.7396511
5	1.0173393	1.5726778	3.6529685	0.6079240	0.8677396
6	1.0173393	1.5726778	3.6529685	0.6079240	0.8677396

Table 12
Numerical data in Example 3 when $\widehat{\lambda} = 15$ per second.

i	$\widehat{\lambda}_i$	s_i	P_i	ρ	T_i
1	2.5417158	1.6389749	4.0884348	0.9088610	2.2205668
2	2.5417158	1.6389749	4.0884348	0.9088610	2.2205668
3	2.4752352	1.6541260	4.1925318	0.8864695	2.6525934
4	2.4752352	1.6541260	4.1925318	0.8864695	2.6525934
5	2.4830422	1.6578105	4.2181080	0.8861475	2.7558910
6	2.4830422	1.6578105	4.2181080	0.8861475	2.7558910

Table 13
The input parameters in Example 4.

i	1	2	3	4	5	6	7	8	9
$\widetilde{\lambda}_i$	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
$\widetilde{r}_i$	0.2	0.3	0.3	0.2	0.3	0.3	0.2	0.3	0.3
α_i	2.8	2.8	2.85	2.8	2.8	2.85	2.8	2.8	2.85
Type	D1	D1	D1	D2	D2	D2	D3	D3	D3

Table 14
Numerical data in Example 4.

i	$\widehat{\lambda}_i$	s_i	P_i	ρ	T_i
1	2.3586593	1.6868677	4.4234147	0.7265126	0.6815649
2	1.7094800	1.6200410	3.9607605	0.7396837	0.7860181
3	1.5924250	1.5677335	3.7018526	0.7382305	0.8062289
4	2.1778665	1.6805501	4.3782302	0.6915909	0.7307691
5	1.3589734	1.5829530	3.7183496	0.6795152	0.9159103
6	1.2196821	1.5219815	3.4102952	0.6747051	0.9516179
7	2.1879243	1.6954609	4.4853658	0.6875850	0.7773529
8	1.2849491	1.5729938	3.6549676	0.6673466	1.0224255
9	1.1100148	1.4984927	3.2667644	0.6596662	1.0791154

where the priority discipline of nodes $v_1, v_2, \ldots, v_3$ is Discipline 1 (abbreviated as D1), the priority discipline of nodes $v_4, v_5, \ldots, v_6$ is Discipline 2 (abbreviated as D2) and the priority discipline of nodes $v_7, v_8, \ldots, v_9$ is Discipline 3 (abbreviated as D3). $P_i^* = 0.1$ W for all $1 \le i \le n$. The preloaded dedicated tasks and α_i of each node are shown in Table 13. The arrival rate of general task is $\widehat{\lambda} = 15$ per second, and its tasks execution requirement is $\widehat{r} = 0.35$ (giga instructions). We assume that we are given $\overline{P} = 35$ W. In Table 14, we show the optimal load distribution, node speed, node utilization and the average general task response time of each node. The overall average task response time of the n nodes is $T = 0.8565089$ s.

Example 5. In this example, we consider a small $\widehat{\lambda}$ under the same situation as in Example 4. We demonstrate the same data with the same input as in Example 4 except that general task arrival rate is $\widehat{\lambda} = 8$ per second. The result is shown in Table 15. The overall average task response time of the n nodes is $T = 0.4940509$ s.

We draw the following important observations from our experimental data.

Table 15
Numerical data in Example 5.

i	$\widehat{\lambda}_i$	s_i	P_i	ρ	T_i
1	1.6501422	1.7133585	4.6162202	0.5705459	0.4280746
2	1.0466706	1.6373952	4.0776802	0.5901658	0.4942597
3	0.9410390	1.5814358	3.7922998	0.5876708	0.5076584
4	1.4230518	1.7031233	4.5410849	0.5273065	0.4479289
5	0.6273117	1.5830631	3.7190541	0.5177046	0.5632899
6	0.4992032	1.5132609	3.3565251	0.5119547	0.5888270
7	1.3626619	1.7188485	4.6568559	0.5101855	0.4877666
8	0.3802008	1.5528580	3.5290115	0.4720781	0.6573717
9	0.0697183	1.4004384	2.7112739	0.4458613	0.7407852

- As shown in Tables 3, 4, and 5, for any two nodes with the same priority discipline and α, the node with a smaller amount of preloaded tasks will be assigned with a greater amount of generic tasks and power.
- As shown in Tables 7, 8, and 9, for any two nodes with the same priority discipline and amount of preloaded tasks, the node with smaller α will be assigned with more tasks and more power.
- As shown in Tables 11 and 12, for any two nodes with the same priority discipline, α, and amount of preloaded tasks, the two nodes will be assigned with the same amount of generic tasks and power.
- As shown in Tables 14 and 15, for any two nodes with the same α and amount of preloaded tasks, the utilization of the node with more urgent tasks will be lower.
- As shown in Tables 11, 12, 14, and 15, in the case of heterogeneous priority of nodes, for any two nodes with the same α and amount of preloaded tasks, the node with priority Discipline 1 will be assigned with more tasks than the node with priority Discipline 2 or Discipline 3, but the node with priority Discipline 2 and the node with priority Discipline 3 can possibly be assigned with almost the same tasks size.

We will plot the average task response time T as a function of $\widehat{\lambda}$. Thus, we need to know the maximum general task arrival rate $\widehat{\lambda}_{max}$ that the system can handle. For any node type, we always have

$$\lim_{\phi\to\infty}\widehat{\lambda}_i = \lim_{\phi\to\infty} f_i(\phi, s_i) = \frac{s_i - \widetilde{\lambda}_i\widetilde{r}_i}{\widehat{r}},$$

where $1 \le i \le n$. Then we have

$$\lim_{\phi\to\infty} g_i\left(\widehat{\lambda}_i, s_i\right) \to \infty,$$

where $1 \le i \le n$. This equation indicates that there always exists a common τ that can satisfy the power constraint when ϕ is large enough. However the given power $\overline{P}$ is limited; therefore, to obtain $\widehat{\lambda}_{max}$, we just need to set ϕ large enough. Then, Algorithm 3 can be used to obtain an approximate $\widehat{\lambda}_{max}$. The average task response time is related to the preloaded tasks, α_i and node queueing discipline. We use the same preloaded tasks, α_i and node queueing discipline in Example 3. Fig. 5 shows the average task response time T as a function of $\widehat{\lambda}$.

As shown in Fig. 5, the average response time T will increase exponentially when the arrival rate $\widehat{\lambda}$ of generic tasks will be larger, and it also shows that our algorithm is feasible.

We also plot the node utilization of each node ρ_i as a function of $\widehat{\lambda}$ (Fig. 6), the power allocation P_i as a function of $\widehat{\lambda}$ (Fig. 7), and the task assignment $\widehat{\lambda}_i$ as a function $\widehat{\lambda}$ (Fig. 8). We assume that the given power is 30 W when these functions are plotted. Fig. 6 shows that as $\widehat{\lambda}$ increases, the difference between the utilizations of each node decreases. From Fig. 7, we know that the power allocated to each node tends to stabilize as $\widehat{\lambda}$ increases, and the difference between the power allocated to each node becomes narrow. Fig. 8 reveals that the key factors affecting the task assignment $\widehat{\lambda}_i$ are the preloaded tasks and α_i, not priority discipline, when $\widehat{\lambda}$ is very large.

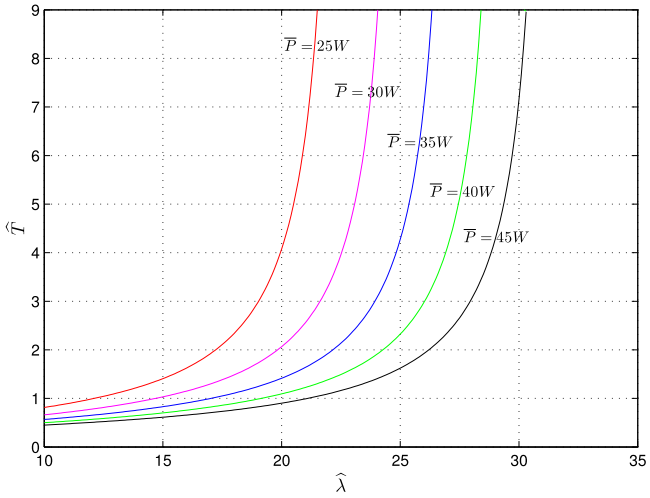

Fig. 5. Average task response time T vs. $\widehat{\lambda}$ and $\overline{P}$.

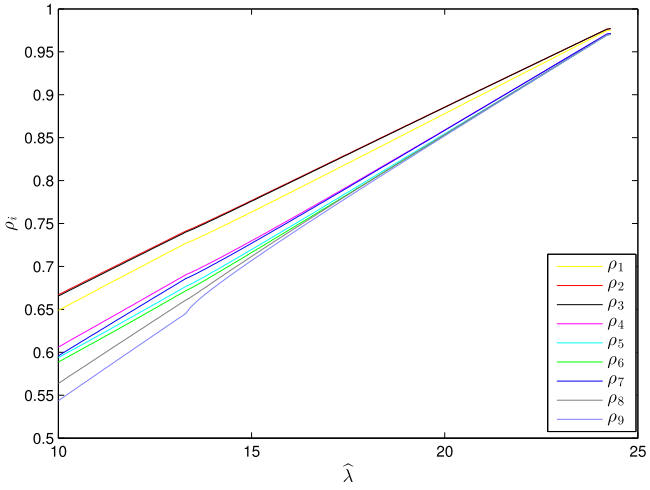

Fig. 6. Node utilization ρ_i vs. $\widehat{\lambda}$.

6. Conclusions

We have mentioned the necessity of heterogeneous distributed-computing architecture for the development of embedded computing, as well as the importance and significance of performance optimization and power reduction in an embedded and distributed environment. We described a queueing model for a group of complete heterogeneous computing nodes with different speeds and an energy-consumption model. We presented a power-allocation and load-balancing problem under the condition that each node employs different queueing disciplines. We constructed our solution as a multi-variable and multi-constrained problem that we addressed by utilizing the Lagrange multipliers and binary search approach, which are demonstrated with some numerical examples. Our work makes an initial contribution to optimal load balancing with a dynamic power constraint for multiple queueing systems with multiple embedded computing nodes in heterogeneous and distributed embedded systems.

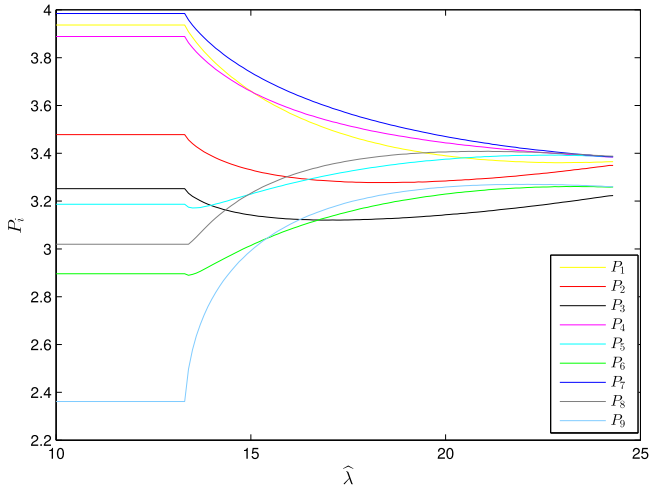

Fig. 7. Power assignment P_i vs. $\widehat{\lambda}$.

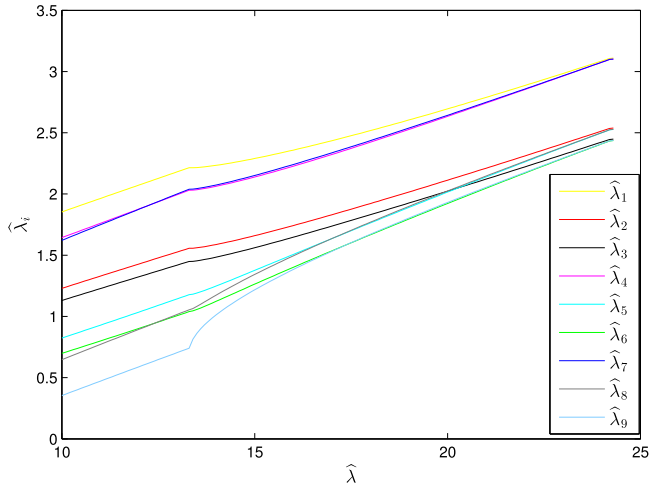

Fig. 8. Generic task assignment $\widehat{\lambda}_i$ vs. $\widehat{\lambda}$.

Acknowledgements

The authors are grateful to the anonymous reviewers for their suggestions to improve the manuscript. This work was supported by the National Natural Science Foundation of China (Grant No. 61872135, 61300037), the Natural Science Foundation of Hunan Province (Grant No. 2018JJ2066).

References

[1] S. Albers, Energy-efficient algorithms, Commun. ACM 53 (5) (2010) 86–96.
[2] A.O. Allen, Probability, Statistics, and Queueing Theory with Computer Science Applications, Academic Press, 1990.
[3] J. Balasangameshwara, N. Raju, A hybrid policy for fault tolerant load balancing in grid computing environments, J. Netw. Comput. Appl. 35 (1) (2012) 412–422.
[4] L.A. Barroso, U. Hölzle, The case for energy-proportional computing, Computer 40 (12) (2007) 33–37.
[5] L. Benini, A. Bogliolo, G. De Micheli, A survey of design techniques for system-level dynamic power management, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 8 (3) (2000) 299–316.
[6] F. Bonomi, A. Kumar, Adaptive optimal load balancing in a nonhomogeneous multiserver system with a central job scheduler, IEEE Trans. Comput. 39 (10) (1990) 1232–1250.
[7] J. Cao, K. Li, I. Stojmenovic, Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers, IEEE Trans. Comput. 63 (1) (2014) 45–58.
[8] M. Guzek, P. Bouvry, E.-G. Talbi, A survey of evolutionary computation for resource management of processing in cloud computing [review article], IEEE Comput. Intell. Mag. 10 (2) (2015) 53–67.
[9] L. He, S.A. Jarvis, D.P. Spooner, H. Jiang, D.N. Dillenberger, G.R. Nudd, Allocating non-real-time and soft real-time jobs in multiclusters, IEEE Trans. Parallel Distrib. Syst. 17 (2) (2006) 99–112.
[10] S.U. Khan, I. Ahmad, A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids, IEEE Trans. Parallel Distrib. Syst. 20 (3) (2009) 346–360.
[11] F. Kong, X. Liu, A survey on green-energy-aware power management for datacenters, ACM Comput. Surv. 47 (2) (2015) 30.
[12] K. Li, Optimal load distribution in nondedicated heterogeneous cluster and grid computing environments, J. Syst. Archit. 54 (1–2) (1991) 111–123.
[13] K. Li, Optimizing average job response time via decentralized probabilistic job dispatching in heterogeneous multiple computer systems, Comput. J. 41 (4) (1998) 223–230.
[14] K. Li, Optimal power allocation among multiple heterogeneous servers in a data center, Sustain. Comput.: Inform. Syst. 2 (1) (2012) 13–22.
[15] K. Li, Optimal load distribution for multiple heterogeneous blade servers in a cloud computing environment, J. Grid Comput. 11 (1) (2013) 27–46.
[16] K. Li, Improving multicore server performance and reducing energy consumption by workload dependent dynamic power management, IEEE Trans. Cloud Comput. 4 (2) (2016) 122–137.
[17] K. Li, Optimal task dispatching on multiple heterogeneous multiserver systems with dynamic speed and power management, IEEE Trans. Sustain. Comput. 2 (2) (2017) 167–182.
[18] K. Li, Quantitative modeling and analytical calculation of elasticity in cloud computing, IEEE Trans. Cloud Comput. PP (99) (2017) 1–1.
[19] Q. Qiu, M. Pedram, Dynamic power management based on continuous-time markov decision processes, in: Proceedings of the 36th Annual ACM/IEEE Design Automation Conference, ACM, 1999, pp. 555–561.
[20] A. Rahman, X. Liu, F. Kong, A survey on geographic load balancing based data center power management in the smart grid environment, IEEE Commun. Surv. Tutor. 16 (1) (2014) 214–233.
[21] C. Rommen, The probability of load balancing success in a homogeneous network, IEEE Trans. Softw. Eng. 17 (9) (1991) 922–933.
[22] K.W. Ross, D.D. Yao, Optimal load balancing and scheduling in a distributed computer system, J. ACM 38 (3) (1991) 676–689.
[23] M.T. Schmitz, B.M. Al-Hashimi, P. Eles, System-Level Design Techniques for Energy-Efficient Embedded Systems, Springer Science & Business Media, 2004.
[24] F. Shengjin, A new extracting formula and a new distinguishing means on the one variable cubic equation, Nat. Sci. J. Hainan Teach. Coll. 2 (2) (1989) 91–398.
[25] B.A. Shirazi, K.M. Kavi, A.R. Hurson, Scheduling and Load Balancing in Parallel and Distributed Systems, IEEE Computer Society Press, 1995.
[26] T. Simunic, L. Benini, G. De Micheli, Energy-efficient design of battery-powered embedded systems, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 9 (1) (2001) 15–28.
[27] M.R. Stan, K. Skadron, Guest editors' introduction: power-aware computing, Computer 36 (12) (2003) 35–38.
[28] X. Tang, S.T. Chanson, Optimizing static job scheduling in a network of heterogeneous computers, in: Parallel Processing, 2000. Proceedings. 2000 International Conference on, 2000, pp. 373–382.
[29] Y. Tian, C. Lin, K. Li, Managing performance and power consumption tradeoff for multiple heterogeneous servers in cloud computing, Clust. Comput. 17 (3) (2014) 943–955.
[30] T.-H. Tsai, L.-F. Fan, Y.-S. Chen, T. Yao, Triple speed: energy-aware real-time task synchronization in homogeneous multi-core systems, IEEE Trans. Comput. 65 (4) (2016) 1297–1309.
[31] L. Wang, S.U. Khan, D. Chen, J. Kołodziej, R. Ranjan, C.-Z. Xu, A. Zomaya, Energy-aware parallel task scheduling in a cluster, Future Gener. Comput. Syst. 29 (7) (2013) 1661–1670.
[32] M. Weiser, B. Welch, A. Demers, S. Shenker, Scheduling for reduced cpu energy, in: Mobile Computing, Springer, 1994, pp. 449–471.
[33] A. Wierman, L.L. Andrew, A. Tang, Power-aware speed scaling in processor sharing systems, in: INFOCOM, IEEE, 2009, pp. 2007–2015.
[34] L. Xiao, S. Boyd, S.-J. Kim, Distributed average consensus with least-mean-square deviation, J. Parallel Distrib. Comput. 67 (1) (2007) 33–46.
[35] G. Xie, G. Zeng, Z. Li, R. Li, K. Li, Adaptive dynamic scheduling on multifunctional mixed-criticality automotive cyber-physical systems, IEEE Trans. Veh. Technol. 66 (8) (2017) 6676–6692.

[36] G. Xie, G. Zeng, Y. Liu, J. Zhou, R. Li, K. Li, Fast functional safety verification for distributed automotive applications during early design phase, IEEE Trans. Ind. Electron. 65 (5) (2018) 4378–4391.
[37] B. Yang, Z. Li, S. Chen, T. Wang, Stackelberg game approach for energy-aware resource allocation in data centers, IEEE Trans. Parallel Distrib. Syst. 27 (12) (2016) 1–1.
[38] B. Zhai, D. Blaauw, D. Sylvester, K. Flautner, Theoretical and practical limits of dynamic voltage scaling, in: Design Automation Conference, DAC 2004, San Diego, CA, USA, June, 2004, pp. 868–873.
[39] X. Zheng, Y. Cai, Optimal server provisioning and frequency adjustment in server clusters, in: Parallel Processing Workshops (ICPPW), 2010 39th International Conference on, IEEE, 2010, pp. 504–511.

Jing Huang received his Ph.D. degree in Computer Science and Technology from Hunan University, Changsha, China, in 2018. He is working as a Postdoctoral Researcher at Hunan University. His research interests include parallel computing, high-performance computing, distributed computing, energy-efficient computing, heterogeneous computing, cloud computing, machine learning.

Yan Liu received the PhD degree in computer science and technology from Hunan University, China, 2010. He is an associate professor at the College of Computer Science and Electronic Engineering of Hunan University, China. His research areas include parallel and distributed system and embedded system.

Renfa Li is a Professor of computer science and electronic engineering, and the Dean of College of Computer Science and Electronic Engineering, Hunan University, China. He is the Director of the Key Laboratory for Embedded and Network Computing of Hunan Province, China. His major interests include computer architectures, embedded computing systems, cyber–physical systems, and Internet of things. He is a member of the council of CCF, a senior member of IEEE, and a senior member of ACM.

Keqin Li is a SUNY Distinguished Professor of computer science with the State University of New York. He is also a Distinguished Professor at Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has published over 630 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He currently serves or has served on the editorial boards of the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Computers, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Services Computing, and the IEEE Transactions on Sustainable Computing. He is an IEEE Fellow.

Jiyao An received the M.Sc. degree in Mathematics from Xiangtan University, China, the Ph.D. degree in Mechanical Engineering from Hunan University, China, in 1998, and 2012, respectively. He was a visiting scholar with the Department of Applied Mathematics, University of Waterloo, Ontario, Canada, from 2013 to 2014. Since 2000, he joined the College of Computer Science and Electronic Engineering in Hunan University, Changsha, China, where he is currently an Professor. His research interests include cyber–physical systems (CPS), Takagi–Sugeno fuzzy systems, parallel and distributed computing, and computing intelligence. He has publish more than 50 papers in international and domestic journals and refereed conference papers. He is a member of the IEEE and ACM, and a senior member of CCF. He is an active reviewer of international journals.

Yang Bai received her B.S. and M.S. degrees from Hunan University in 2013 and 2016, respectively. She is currently working on the Ph.D. degree at Hunan Province, Hunan University. Her research interests include service computing, embedded systems, cyber–physical systems.

Fan Yang received the PhD degree in computer science and technology from Hunan University, China, in 2016. He is currently an assistant professor in Central South University of Forestry and Technology.He was a visit scholar with Michigan State University, from 2014-2015. His research interests include cyber-physical systems, embedded systems, and modeling. He is a member of China Computer Federation.

Guoqi Xie received his Ph.D. degree in computer science from Hunan University, China, in 2014. He was a postdoctoral researcher at Nagoya University, Japan, from 2014 to 2015. Since 2015 he is working as a Postdoctoral Researcher at Hunan University, China, since 2015. His major interests include embedded systems, distributed systems, real-time systems, in-vehicle networks, and cyber–physical systems. He is a member of IEEE and ACM.