A Survey of Low-Energy Parallel Scheduling Algorithms

Guoqi Xie, *Senior Member, IEEE*, Xiongren Xiao, Hao Peng, Renfa Li, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

Abstract—High energy consumption is one of the biggest obstacles to the rapid development of computing systems, and reducing energy consumption is quite urgent and necessary for sustainable computing. Low-energy scheduling based on dynamic voltage and frequency scaling (DVFS) is one of the most commonly used energy optimization techniques. Recent survey works have reviewed some low-energy scheduling algorithms, but there is currently no systematic review in low-energy *parallel* scheduling algorithms. With the increasing complexity of function requirements, many parallel applications have been executed in various sustainable computing systems. In this paper, we survey recent advances in low-energy parallel scheduling algorithms according to three scheduling styles, namely: 1) energy-efficient parallel scheduling algorithms; 2) energy-aware parallel scheduling algorithms; and 3) energy-conscious parallel scheduling algorithms. Low-energy parallel scheduling algorithms basically involve five categories of 1) heuristic algorithms; 2) meta-heuristic algorithms; 3) integer programming algorithms; 4) machine learning algorithms; and 5) game theory algorithms. Further, we introduce the future trends in low-energy parallel scheduling algorithms from the perspectives of new requirements and future developments. By surveying the recent advances and introducing the future trends, we expect to provide researchers with a systematic reference and development directions in low-energy parallel scheduling for sustainable computing systems.

Index Terms—Energy-aware, energy-conscious, energy-efficient, parallel scheduling

1 Introduction

1.1 High Energy Consumption of Computing Systems

Since the birth of the computer, people have never stopped the pursuing of low energy consumption for sustainable computing systems. The reason is that high energy consumption brings the following problems.

1) High temperature problem. Fig. 1 shows the power consumptions of modern multi-core processors under idle and multi-thread conditions. From Fig. 1, we can see that the power of the latest multi-core processor is as high as 70w under idle condition and even more than 200w under multi-threaded condition [1]. At present, some physical temperature cooling technologies, including water cooling, air cooling, evaporative cooling, and their combinations, are all used to cool computing systems.
2) Electromagnetic interference problem. High energy consumption generates strong electromagnetic radiation, which will make the surrounding equipment malfunction, thereby causing unnecessary safety issues [2].
3) Environmental deterioration problem. High energy consumption affects battery life and causes environmental deterioration problem. Based on the strategy of advocating green computing and environmental protection, people increasingly support sustainable computing systems [3].
4) Global economic problem. Huge energy consumption is generated by clusters, supercomputers, and cloud data centers, which have powerful computing and storage capabilities but generate huge energy consumption. The National Natural Resources Defense Council (NRDC) survey report shows that the energy consumption of data centers is expected to increase to about 140 billion kWh per year by 2020 [4]. The current global data center energy consumption accounts for about 5 percent of global energy consumption.

As can be seen from the above problems, high energy consumption is one of the biggest obstacles to the rapid development of sustainable computing systems, and reducing energy consumption is quite urgent and necessary.

1.2 Low-Energy Scheduling

To effectively reduce energy consumption from the operating system level, various low-energy scheduling algorithms have been proposed. Low-energy scheduling is a branch of scheduling, and it is a common system-level energy optimization technique [5]. When designing low-energy scheduling algorithms, Dynamic Voltage and Frequency Scaling (DVFS) is one of the most commonly used techniques in sustainable computing systems [6]. By adjusting the voltage and frequency of the

• *Guoqi Xie, Xiongren Xiao, Hao Peng, and Renfa Li are with the Key Laboratory for Embedded and Cyber-Physical Systems of Hunan Province, College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410082, China. E-mail: {xgqman, xxr, hao_peng, lirenfa}@hnu.edu.cn.*
• *Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA. E-mail: lik@newpaltz.edu.*

Manuscript received 2 Sept. 2020; revised 13 Dec. 2020; accepted 5 Feb. 2021. Date of publication 9 Feb. 2021; date of current version 7 Mar. 2022. (Corresponding authors: Xiongren Xiao, Renfa Li, and Keqin Li.) Recommended for acceptance by V. Eramo. Digital Object Identifier no. 10.1109/TSUSC.2021.3057983

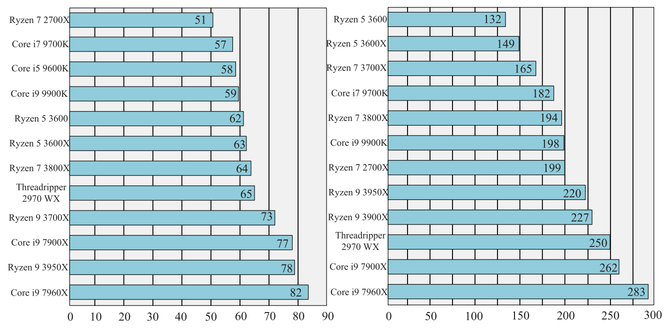


Fig. 1. Power consumptions of Modern multi-core processors.

processor, both the system performance and energy consumption can be dynamically adjusted (the details of DVFS can be found in Section 2). Currently, most commercial processors produced by chip manufacturers (e.g., Intel [7], AMD [8], ARM [9], etc.) support DVFS, and corresponding software systems (e.g., the CPUFreq subsystem [9]) are developed to support core frequency adjustment.

Low-energy scheduling algorithms based on DVFS generally include three styles: 1) energy-efficient scheduling algorithms; 2) energy-aware scheduling algorithms; and 3) energy-conscious scheduling algorithms.

1) Energy-efficient scheduling algorithms aim to reduce the energy consumption in an efficient way while meeting one or multiple requirements (e.g., response time requirement, reliability requirement, cost requirement, and security requirement etc.).
2) Energy-aware scheduling algorithms aim to improve the performance (e.g., improve the reliability, shorten the response time, reduce the cost, and enhance the security etc.) as much as possible while meeting the energy consumption constraint.
3) Energy-conscious scheduling algorithms aim to simultaneously optimize energy consumption and other multiple performances metrics (e.g., high reliability, short response time, and low cost etc.), thereby sufficiently improving the operating efficiency of sustainable computing systems.

1.3 Application-Oriented Sustainable Computing Systems

The computing system based on von Neumann architecture have remained essentially unchanged for decades. However, through the continuous penetration of information technology, computing systems have developed into application-oriented computing systems, such as Cyber-Physical Systems (CPS), Internet of Things (IoT), Cloud Computing Systems (CCS), etc., where low-energy parallel scheduling algorithms play an important role in energy saving. CPS integrate advanced sensing, computing, communication, and control technologies to implement the dynamic interaction process between cyber systems and physical systems [10]. IoT utilizes the ubiquitous information interaction and communication of smart and mobile devices to realize the seamless connection between persons and things through advanced network technologies [11]. CCS provide Quality of Services (QoS) for users by pooling and allocating configurable resources (including servers, networks, storage, etc.) [12]. Other application-oriented computing systems, such as embedded, mobile, cluster, grid, edge-cloud systems, etc., may also require low-energy parallel scheduling algorithms.

(1) Safety-critical systems require energy-efficient scheduling algorithms to reduce the energy consumption while meeting the safety requirement. Safety-critical systems mean that the safety of the system is quite important, and the safety requirement must be met to ensure normal execution; if the designed system cannot meet the certified safety requirement, it may cause serious consequences (including property damage and even casualties etc.) during actual operation. For example, the safety problem caused by the malfunction in automotive CPS may result in property damage and life-threatening injuries [13]. The brake-by-wire application in automotive CPS must perform its braking action correctly in the specified real-time requirement (i.e., deadline), which is an important safety requirement; otherwise, a collision may occur if the actual response time exceeds its deadline, resulting in injury to drivers, passengers, and pedestrians. Therefore, meeting the safety requirement is a prerequisite for normal operation of CPS, and then energy-efficient scheduling algorithms is employed to reduce the energy consumption.

(2) Lifetime-limited systems require energy-aware scheduling algorithms to increase the performance while meeting the energy consumption constraint. For instance, IoT devices are lifetime-limited systems because they usually use battery-powered systems, such that energy provision is limited [14]. IoT devices are widely employed in industrial production systems (e.g., smart manufacturing and smart factory). Tens of thousands of IoT devices and edge computing servers are deeply connected and integrated through the industrial networks [15]. Meanwhile, as the scale of IoT continues to increase, the application computing workloads undertaken by edge computing servers and IoT devices are increasingly heavy, such that high-performance industrial IoT devices and edge computing servers are required to provide high-performance computing environments.

(3) High-performance systems require energy-conscious scheduling algorithms to implement the joint optimization of energy consumption reduction and other QoS improvement. For instance, CCS are high-performance computing systems, which can provide strong computational services for users in the form of fixed rent (i.e., short-term lease) or pay-as-you-go (i.e., short-term lease) [16]; meanwhile, CCS provide multiple QoS metrics to users through on-demand computational services. One of the core competitiveness of CCS service providers is the ability to efficiently provide these QoS metrics and reduce energy consumption.

Note that energy-efficient scheduling, energy-aware scheduling, and energy-conscious scheduling are not exclusively targeted to specific platforms (e.g., CPS, IoT, and CCS). For instance, most effort concerns stream processing and fog-edge-cloud interaction in IoT, and significant effort was devoted to server consolidation in CCS. Furthermore, deadline-aware (i.e., real-time) low-energy parallel scheduling appeared as a problem well before the CPS and IoT era [21], [22]. For instance, it is understandable that deadline-aware scheduling fits both CPS and cloud processing of scientific workflows. Further, it is feasible to perform multi-objective

TABLE 1
Recent Survey Works have Reviewed Some Low-Energy Scheduling Algorithms

Reference	Year	Surveyed algorithms
[17]	2014	Heuristics, game theory, and machine learning
[18]	2016	Algorithms for single processor and multiprocessors
[19]	2016	Heuristic algorithms with the multi-objective problems
[12]	2019	Meta-heuristic algorithms considering virtualization, consolidation, and energy-awareness
[20]	2020	Machine learning algorithms from optimization goal, optimization technique, and system architecture

optimization with energy as one of the optimization functions in the case of workflow scheduling in IoT.

1.4 Recent Survey Works

Recent survey works have reviewed some low-energy scheduling algorithms, as shown in Table 1. Ref. [17] provided a short survey in low-energy parallel scheduling algorithms of heuristics, game theory, and machine learning. Ref. [18] reviewed the low-energy scheduling algorithms (including single processor and multiprocessors) from 1990 to 2016. Ref. [19] took performance, energy and temperature as optimization objectives, and it summarized 16 heuristic low-energy scheduling algorithms to address the multi-objective problems. Ref. [12] summarized the meta-heuristic algorithms widely used in low-energy scheduling through three dimensions (i.e., virtualization, consolidation, and energy-awareness). Ref. [20] is the latest overview of machine learning algorithms for low-energy scheduling from multiple dimensions (including optimization goal, optimization technique, and system architecture). However, there is currently no systematic review of low-energy parallel scheduling algorithms.

Low-energy parallel scheduling means low-energy scheduling for parallel applications. From a perspective of scheduling, a parallel application comprises multiple tasks (processes), where some tasks could be executed in parallel in multiple processors (or cores), and some tasks exist data dependency and precedence constraints; through inter-core data exchange or network communication transmission, a task can send data to its immediate successor tasks to trigger the execution of these tasks. In general, a parallel application could be denoted with a Directed Acyclic Graph (DAG) [23], where a node in a DAG represents a computing task, and each edge between two nodes represents their precedence constraint. In addition, Task Interaction Graph (TIG) [24], hybrid DAG (HDAG) [24], Precedence-constrained Task Graph (PTG) [25], and Activity on Edge (AOE) network [26], etc. are DAG variants, and they can also be used to represent a parallel application.

With the increasing complexity of function requirements, many parallel applications have been executed in in sustainable computing systems, such as CPS, IoT and CCS. For instance, Adaptive Cruise Control (ACC) and brake-by-wire are parallel applications in automotive CPS; Random Linear Network Coding (RLNC) is a parallel application in IoT devices [27]; and scientific workflows (e.g., Montage, CyberShake, Epigenomics etc.) are parallel applications in CCS [28], [29]. These applications are designed to make full use of multi-core, multi-processor, or multi-machine architecture to improve system efficiency.

Based on the above research background, Fig. 2 shows the overview of recent advances in low-energy parallel scheduling algorithms in this paper.

1.5 Our Contributions

At present, there is no investigation to classify low-energy parallel scheduling algorithms according to multiple scheduling styles, and multiple algorithm categories. For the research of the investigation of low-energy parallel scheduling algorithms, we make the following contributions.

1) We survey the recent advances in low-energy parallel scheduling algorithms according to three scheduling styles, namely, 1) energy-efficient parallel scheduling algorithms; 2) energy-aware parallel scheduling algorithms for; and 3) energy-conscious parallel scheduling algorithms. Low-energy parallel scheduling algorithms basically involve five categories of 1) heuristic algorithms; 2) meta-heuristic algorithms; 3) integer programming algorithms; 4) machine learning algorithms; 5) and game theory algorithms.
2) We introduce the future trends in low-energy parallel scheduling algorithms from the perspectives of new requirements and future developments.

By surveying the recent advances and introducing the future trends, we hope to provide researchers with a systematic reference and development directions in low-energy parallel scheduling.

The rest of this work is summarized below. Section 2 gives preliminaries related to low-energy parallel scheduling. Sections 3, 4, and 5 survey the recent advances in energy-efficient parallel scheduling algorithms, energy-aware parallel scheduling algorithms, and energy-conscious parallel scheduling algorithms, respectively. Section 6 introduces the future trends in low-energy parallel scheduling algorithms. The paper is concluded in Section 7.

2 Architectures and Models

2.1 Computing System Architectures

The low-energy parallel scheduling algorithms depend on the system architecture of specific domains. This paper uses CPS, IoT and CCS as examples to introduce their system architectures.

1) The architecture of CPS is usually a heterogeneous networked multi-processor architecture, where processors are scattered in different locations but interconnected via various network buses. Fig. 3 shows the networked architecture of automotive CPS, where industrial Ethernet, Control Area Network

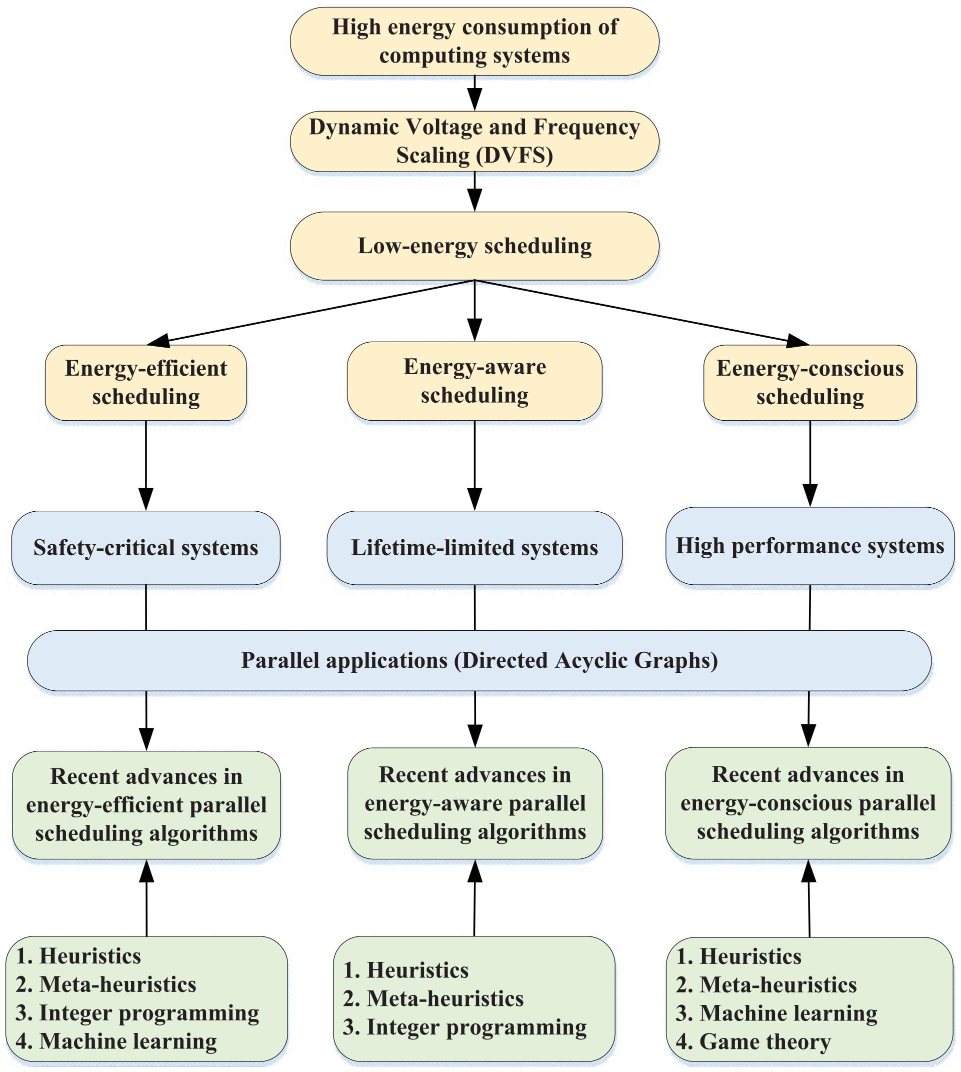

High energy consumption of computing systems
Dynamic Voltage and Frequency Scaling (DVFS)
Low-energy scheduling
Energy-efficient scheduling
Energy-aware scheduling
Eenergy-conscious scheduling
Safety-critical systems
Lifetime-limited systems
High performance systems
Parallel applications (Directed Acyclic Graphs)
Recent advances in energy-efficient parallel scheduling algorithms
Recent advances in energy-aware parallel scheduling algorithms
Recent advances in energy-conscious parallel scheduling algorithms
1. Heuristics 2. Meta-heuristics 3. Integer programming 4. Machine learning
1. Heuristics 2. Meta-heuristics 3. Integer programming
1. Heuristics 2. Meta-heuristics 3. Machine learning 4. Game theory

Fig. 2. Overview of recent advances in low-energy parallel scheduling algorithms in this survey.

(CAN), and FlexRay with different bandwidths are used to construct an in-vehicle network and connect all processors.

2) The architecture of IoT devices is usually a heterogeneous multiprocessor System-on-Chip (MPSoC) architecture, where multiple processor cores with different instruction sets are integrated in a single chip. However, the increase in the number of cores/processors generates further data transmission and affects the performance of IoT devices. Currently, several hardware techniques (e.g., big. LITTLE core architecture) are proposed to reduce data transmission. Fig. 4 shows a MPSoC architecture (i.e., Zynq Ultrascale + MPSoC) that usually used in IoT devices.

3) The architecture of CCS is usually a heterogeneous distributed multiple Virtual Machines (VM) architecture, where the hosts of VMs are the servers scattered in different physical locations. The servers interact with each other through a local area networks or Internet, as shown in Fig. 5. In the life cycle of CCS, the old and low-performance servers are always replaced by new high-performance servers to provide scalable computing services.

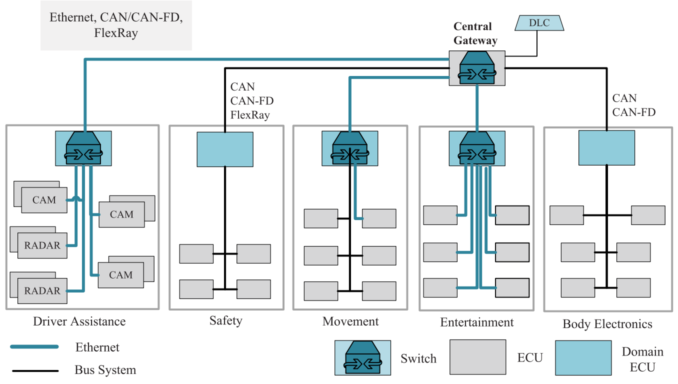


Fig. 3. Heterogeneous distributed multi-processor architecture of automotive CPS [30].

2.2 System-Level Power and Energy Models

There are two main technologies for reducing energy consumption of computing systems: Dynamic Power Management (DPM) and DVFS [32]. DPM mainly switches its working state according to changes in the workload of the

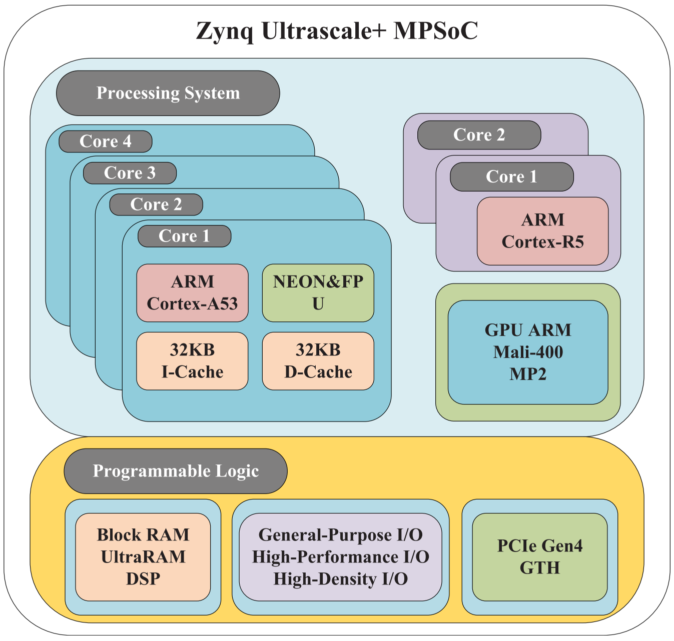


Fig. 4. Heterogeneous MPSoC architecture of IoT devices [31].

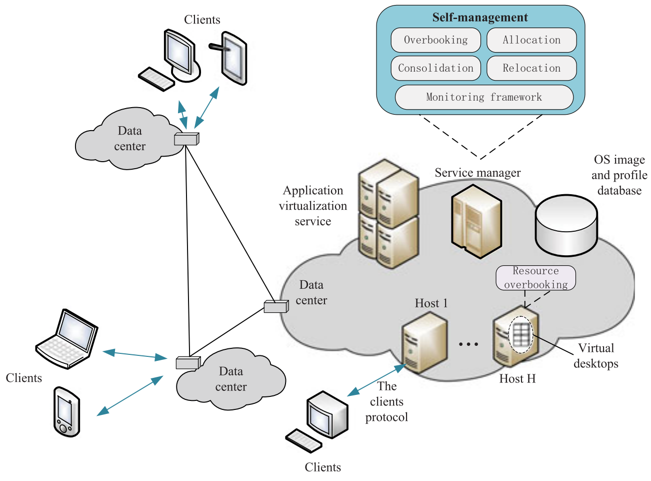


Fig. 5. Heterogeneous distributed multiple VM architecture of CCS.

equipment to minimize the power consumption; for example, when the idle period of the peripheral is long enough, it can be turned off or enters a sleep state to save battery's power. DVFS dynamically adjusts the power supply voltage and clock frequency of the processor to achieve a balance between the response time and energy consumption of the system. Compared with DPM, the main advantage of DVFS is that it can adopt low-energy scheduling algorithms at the operating system level to efficiently use the processor; therefore, we survey low-energy parallel scheduling algorithms that adopt the DVFS technique in this paper. The energy consumption of a computing system can be generated by processors, memory, interface network cards, and external circuit systems etc. Among them, the energy consumption generated by processors is the main part [33].

In the system-level DVFS-enabled power and energy models, the calculation of the total energy consumption of the system is a nonlinear function of the voltage and frequency. In the past, CMOS circuits were usually executed in environments, where voltage is higher than the threshold voltage, and the dynamic energy consumption was the main part of the total energy consumption. With the advancement of Very large-scale integration (VLSI) technology, the integration of transistors has increased while the system size has continued to decrease. Therefore, the gap between the supply and the threshold voltages is narrowed, resulting in the increase of static energy consumption [18]. One of the widely used system-level power model is denoted by [33]

$$P = P_{\mathrm{s}} + \hbar(P_{\mathrm{ind}} + P_{\mathrm{fd}}). \tag{1}$$

The first part of P_{s} represents static power, which can only be eliminated by shutting down the entire system. The second part $\hbar(P_{\mathrm{ind}} + P_{\mathrm{d}})$ represents the dynamic power, where P_{ind} represents the frequency-independent dynamic power and can be eliminated by switching the system to the sleep state; P_{fd} represents the frequency-dependent dynamic power; $\hbar$ represents the state of the system, indicating whether the system currently has dynamic power part. $\hbar = 1$ means that system is active; otherwise, $\hbar = 0$ means that system is not active.

At present, the relevant literature probably provides three models representing frequency-dependent dynamic power P_{fd} and dynamic energy consumption E_{d}.

1) Since DVFS can change the system power by adjusting the voltage and frequency, P_{fd} can be expressed according to voltage and frequency, namely

$$P_{\mathrm{fd}} = C_{\mathrm{ef}} V^2 f. \tag{2}$$

C_{ef} represents the effective capacitance; V represents the supply voltage; C_{ef} is a constant value related to a specific processor, which can be obtained by processor power measurement; f represents the current frequency of the processor. On the basis of Eq. (2), the system-level energy model can be expressed by

$$E_{\mathrm{d}} = \left(P_{\mathrm{ind}} + C_{\mathrm{ef}} V^2 f\right) \times \frac{f_{\max}}{f} \times w. \tag{3}$$

$f_{\max}$ represents the maximum frequency of the processor; V represents the maximum voltage of the system; w is the execution time of a task under the maximum frequency.

2) P_{fd} can also be expressed only in terms of frequency, namely

$$P_{\mathrm{fd}} = C_{\mathrm{ef}} f^m, \tag{4}$$

m, which is a constant value, represents the dynamic power exponent related to a specific processor; m is larger than 2. On the basis of Eq. (4), the system-level energy model can be expressed by

$$E_{\mathrm{d}} = (P_{\mathrm{ind}} + C_{\mathrm{ef}} f^m) \times \frac{f_{\max}}{f} \times w. \tag{5}$$

3) Considering that processor execution speed is usually linear related to frequency, P_{fd} can also be expressed by the processor execution speed s. This model is often adopted when using queuing theory [34], [35], [36] and it is denoted by

$$P_{\text{fd}} = \xi s^{\alpha}. \tag{6}$$

ξ is a constant value that can be deducted from $C_{\text{ef}}V^2 f$; α is the exponent related to processor execution speed and it is also a constant (e.g., constant 3 [37], [38], [39]) related to special processors. On the basis of Eq. (6), the system-level energy model can be expressed by

$$E_{\text{d}} = (P_{\text{ind}} + \xi s^{\alpha}) \times \frac{f_{\max}}{f} \times w. \tag{7}$$

Assume that $E(G)$ is the energy consumption of the application G, then $E(G)$ contains $E_s(G)$ (i.e., static energy consumption) and $E_{\text{d}}(G)$ (i.e., dynamic energy consumption) of the application, namely

$$E(G) = E_s(G) + E_{\text{d}}(G).$$

$E_{\text{s}}(G)$ depends on the makespan (i.e., response time) of the application

$$E_s(G) = \sum_{u_k \in U} P_s \times makespan(G).$$

U represents the processor (or core) set; notice that each processor consumes static energy consumption, such that the static energy consumption of the application contains all static energy consumptions generated in processors.

On the basis of Eq. (3), $E_{\text{d}}(G)$ depends on the processor frequency and is calculated by

$$E_{\text{d}}(G) = \sum_{n_i \in N} \left(\left(P_{\text{ind}} + C_{\text{ef}} V^2 f_i\right) \times \frac{f_{\max}}{f_i} \times w_i \right). \tag{8}$$

f_i represents assigned processor frequency to task n_i; w_i represents the execution time of task n_i under the maximum frequency. N represents the task set of application G; notice that each task generates dynamic energy consumption, such that the dynamic energy consumption of the application contains all the dynamic energy consumptions generated by tasks.

Similar to Eq. (8), other calculations of $E_{\text{d}}(G)$ based on Eqs. (4) and (6) are denoted by

$$E_{\text{d}}(G) = \sum_{n_i \in N} \left(\left(P_{\text{ind}} + C_{\text{ef}} f_i^m\right) \times \frac{f_{\max}}{f_i} \times w_i \right),$$

and

$$E_{\text{d}}(G) = \sum_{n_i \in N} \left((P_{\text{ind}} + \xi s^{\alpha}) \times \frac{f_{\max}}{f_i} \times w_i \right),$$

respectively.

2.3 A Derived Version of System-Level Power and Energy Models

A derived version of the system-level power model is often used in some works [40], [41], [42]. This model contains two parts: 1) idle power (including static power and frequency-independent dynamic power) of processors; 2) frequency-dependent dynamic power of processors; such model is usually used in an application with IO-intensive tasks, and the power calculation is denoted by

$$P = P_{\text{idle}} + P_{\text{fd}} = \beta + kf^{\alpha}. \tag{9}$$

β represents idle power; kf^{α} represents frequency-dependent dynamic power; k and α are energy-efficient capacitors and power exponent, respectively.

On the basis of Eq. (9), the total energy consumption of the application is the sum of idle consumption $E_{\text{idle}}(G)$ and frequency-dependent dynamic energy consumption $E_{\text{fd}}(G)$ of the application, namely

$$E(G) = E_{\text{idle}}(G) + E_{\text{fd}}(G).$$

$E_{\text{idle}}(G)$ depends on $Slack(G)$, which represents the total slack size of the application G; therefore, we have

$$E_{\text{idle}}(G) = \beta \times Slack(G).$$

The frequency-dependent dynamic energy consumption $E_{\text{d}}(G)$ of the application is calculated by

$$E_{\text{fd}}(G) = \sum_{n_i \in N} \left((kf^{\alpha}) \times \frac{f_{\max}}{f_i} \times w_i \right).$$

3 Energy-Efficient Parallel Scheduling Algorithms

In this section, we survey recent advances in energy-efficient parallel scheduling algorithms. Through extensive search and identification, the energy-efficient parallel scheduling algorithms involve four categories: 1) heuristic algorithms; 2) meta-heuristic algorithms; 3) integer programming algorithms; and 4) machine learning algorithms. Unfortunately, there is no research on energy-efficient parallel scheduling based on the game theory algorithm. Game theory embodies a kind of antagonistic game behavior, where each party participating in the competition has individual objective. All parties try to choose the most beneficial plans for them; therefore, game theory is suitable for multi-objective optimization algorithms. Since energy-efficient parallel scheduling is a single-objective optimization problem that only seeks to minimize energy consumption, game theory is not applicable. Fig. 6 surveys the overall algorithms categories of recent advances in energy-efficient parallel scheduling according to heuristics, meta-heuristics, integer programming, and machine learning.

3.1 Heuristic Algorithms

It is an NP-hard optimization problem when finding the optimal solution for energy-efficient parallel scheduling with certain requirement in multiprocessors [43], and even a simple dual-processor scheduling problem with independent tasks is NP-hard. Using heuristic algorithms is a type of solutions that can give a feasible solution in an acceptable time and space when solving a specific problem; however, a heuristic algorithm can neither obtain the optimal solution nor the deviation of the feasible solution from the optimal solution. A heuristic algorithm can change its search path based on individual or global experience in the process of seeking the optimal solution. When the optimal solution to the problem becomes impossible or difficult to complete (e.g., NP-hard optimization problem), heuristic algorithms

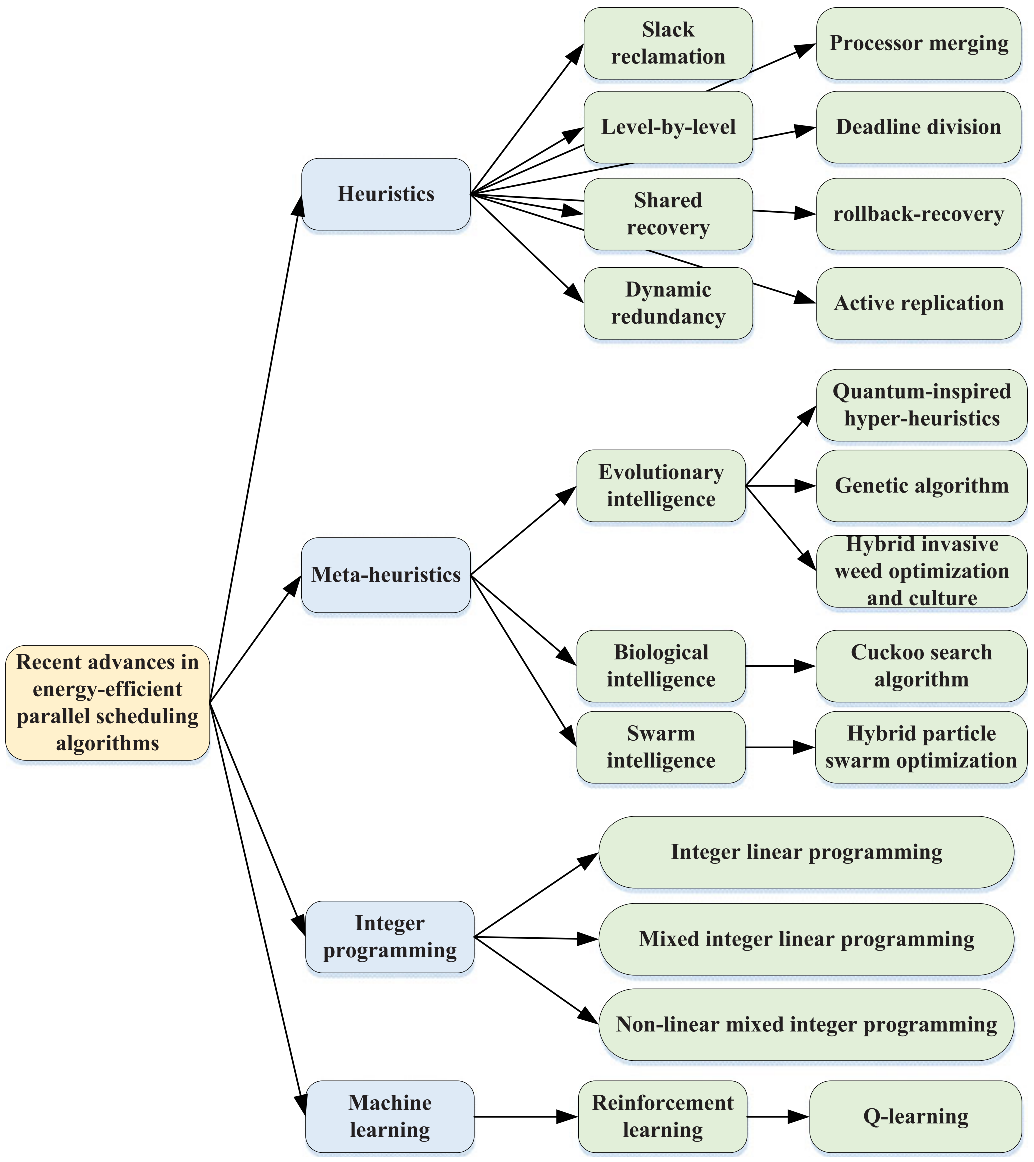


Fig. 6. Overall algorithms categories of recent advances in energy-efficient parallel scheduling according to heuristics, meta-heuristics, integer programming, and machine learning.

are efficient ways to obtain approximate optimal solutions. Due to the simplicity and time efficiency of the heuristic algorithm, it is the most used algorithm among the five categories of algorithms mentioned in this paper.

(1) Energy-efficient parallel scheduling algorithm while meeting the real-time requirement

In Table 2, we survey the recent advances in heuristic energy-efficient parallel scheduling algorithms while meeting the real-time requirement. Real-time requirement means that the application must finish its execution during its deadline, and it is an important safety requirement in time-critical computing systems.

Energy-efficient parallel scheduling while meeting the real-time requirement in homogeneous multiprocessor systems (i.e., homogeneous systems) can be found in Refs. [43], [44], [45]. Ref. [43] addressed the combinatorial optimization problem of energy-efficient parallel scheduling while meeting the real-time requirement in homogeneous systems; to enable heuristic algorithms, the problem is decomposed into three subproblems: 1) precedence constraint; 2) task scheduling; and 3) power supplying. Ref. [44] constructed the level-by-level energy-efficient parallel scheduling algorithm in homogeneous systems; the problem is addressed by using the two-level energy/time/power assignment approach: 1) optimal energy/time assignment to tasks among different levels; 2) equal power provision for tasks of the same level. To address the same problem as Refs. [43], [44], a task scheduling using list scheduling algorithm and speed determination using list placement algorithm were developed in Ref. [45].

Ref. [46] first introduced the upward slack reclamation technique to implement the energy-efficient parallel scheduling algorithm while meeting the real-time requirement in heterogeneous multiprocessor systems (i.e., heterogeneous systems); upward reclamation technique means that reclaiming the slacks from the exit task to the entry task in the parallel application, where tasks are sorted according to the descending order of maximum Latest Finish Time (LFT). Considering that Ref. [46] tried to put all tasks into the lowest frequency to execute, resulting in the overall energy consumption of the application is not reduced enough, Ref. [47] employed the

TABLE 2
Recent Advances in Heuristic Energy-Efficient Parallel Scheduling Algorithms While Meeting the Real-Time Requirement

Reference	Year	Environment	Requirement	Technique	Static/dynamic
[43]	2012	Homogeneous systems	Real-time requirement	Decomposing three subproblems	Static
[44]	2015	Homogeneous systems	Real-time requirement	Level-by-level	Static
[45]	2016	Homogeneous systems	Real-time requirement	List scheduling and list placement	Static
[46]	2012	Heterogeneous systems	Real-time requirement	Upward slack reclamation	Static
[47]	2014	Heterogeneous systems	Real-time requirement	Iterative slack reclamation	Static
[48]	2015	Heterogeneous systems	Real-time requirement	Partial optimal slack reclamation	Static
[49]	2017	Heterogeneous systems	Real-time requirement	Slack reclamation for independent tasks	Static
[50]	2017	Heterogeneous systems	Real-time requirement	Upward and backward slack reclamation	Static
[51]	2017	Heterogeneous systems	Real-time requirement	Global slack reclamation	Static
[52]	2016	Heterogeneous systems	Real-time requirement	Processor merging	Static
[53]	2017	Heterogeneous systems	Real-time requirement	Energy-aware processor merging	Static
[54]	2020	Heterogeneous systems	Real-time requirement	Global processor merging	Static
[55]	2020	Heterogeneous systems	Real-time requirement	Binary processor merging	Static
[56]	2018	Heterogeneous systems	Real-time requirement	Deadline partition for sub-applications	Static
[57]	2018	Homogeneous systems	Real-time requirement	Setting optimal task execution speed	Static
[58]	2019	Heterogeneous systems	Real-time requirement	Deadline partition for tasks	Static
[59]	2020	Heterogeneous systems	Real-time requirement	Dividing deadlines and sorting tasks	Static

improved slack reclamation technique called iterative slack reclamation to achieve the energy-efficient parallel scheduling; the iterative slack reclamation technique iteratively distributes the slack between the tasks and gradually scales rather than directly using the lowest frequency. Considering that Refs. [46], [47] cannot efficiently utilize the slacks in the slack reclamation, an enhanced energy-efficient parallel scheduling algorithm was designed in [48] by using the partial optimal slack reclamation technique, which can make full use of partial slacks. Ref. [49] devised an energy-efficient parallel scheduling algorithm by finding the maximum independent task set of each task, and then iteratively assigning each slack to this set (i.e., the reclamation technique for independent task set), and such technique can maximize the total energy consumption reduction.

Considering that Refs. [46], [47], [48], [49] merely reclaim the slack from the exit task to the entry task, Ref. [50] used the upward-downward slack reclamation technique to achieve the energy-efficient parallel scheduling, which is more energy-efficient than the previous upward slack reclamation techniques. Both the upward and upward-downward slack reclamation techniques belong to the LFT-based slack reclamation approach, which employees the local slack reclamation technique. The local slack reclamation technique means merely reclaiming slacks in the constant processor for each task. To improve the limitation of the local slack reclamation technique, the global slack reclamation technique was adopted in Ref. [51], where tasks can be migrated to the slacks of other processor that can generate the minimum energy consumption while still meeting the real-time requirement of the parallel application; considering that merely using DVFS technique is not enough in practice, Ref. [51] further combined the non-DVFS energy-efficient parallel scheduling and global energy-efficient parallel scheduling to address the above problem.

Processor merging, which powers off certain specific processors, is another important technique to implement the energy-efficient parallel scheduling. The processor merging technique in Ref. [52] powers off processors with a small number of tasks while meeting the real-time requirement of the parallel application. However, neither powering off the processors with a small number of tasks nor powering off as many processors as possible produces a good energy-efficient parallel scheduling. Ref. [53] improved the energy-aware processor merging technique by powering off the most effective processor in terms of saving energy. However, the energy-efficient parallel scheduling algorithms in Refs. [52], [53] are essentially the local processor merging technique (i.e., each task is assigned to a constant processor and can not be migrated). Therefore, Ref. [54] utilized the global processor merging technique to implement the energy-efficient parallel scheduling for multiple real-time parallel applications; this technique can enable tasks migrate to the slacks of other processors. Recently, by analyzing the improved processor merging technique in Ref. [53], a new processor merging technique using the binary search (i.e., the binary processor merging technique) was applied in Ref. [55], which implements the fast energy-efficient parallel scheduling.

Recently, some new energy-efficient parallel scheduling algorithms for specific scenarios were proposed [56], [57], [58], [59], where a core step is deadline division (i.e., deadline partition). Ref. [56] studied energy-efficient parallel scheduling by proposing the deadline division technique, through which sub-deadlines of sub-workflows (i.e., sub-applications) are obtained. Ref. [57] investigated the energy-efficient parallel scheduling by setting optimal task execution speed towards the energy-efficient parallel scheduling when static energy consumption is a major proportion of total energy consumption. Ref. [58] exploited the list-based energy-efficient parallel scheduling algorithm (including top-level task calculation, deadline division for sub-deadlines of tasks, dynamic adjustment, and edge data communication) to reduce the network energy consumption for fat-tree interconnection networks. Ref. [59] provided a two-step (i.e., deadline division and sorting tasks) technique to achieve the energy-efficient parallel scheduling with geo-distributed data (i.e., data is distributed geographically in the data center).

(2) Energy-efficient parallel scheduling while meeting the reliability requirement

Besides the real-time requirement, reliability requirement is another important safety requirement in time-critical

TABLE 3
Recent Advances in Heuristic Energy-Efficient Parallel Scheduling Algorithms While Meeting the Reliability Requirement (Some Works Also Meet the Real-Time Requirement)

Reference	Year	Environment	Requirement	Technique	Static/dynamic
[60]	2013	Uniprocessor systems	Reliability requirement, real-time requirement	Shared recovery	Dynamic
[61]	2018	Heterogeneous systems	Reliability requirement	Checkpointing with rollback-recovery	Dynamic
[62]	2015	Homogeneous systems	Reliability requirement	Dynamic redundancy	Dynamic
[63]	2018	Heterogeneous systems	Reliability requirement, real-time requirement	Active replication	Static
[64]	2019	Heterogeneous systems	Reliability requirement, real-time requirement	Three-stage, active replication	Static

computing systems. In Table 3, we survey the recent advances in heuristic energy-efficient parallel scheduling algorithms while meeting the reliability requirement (some works also meet the real-time requirement).

Ref. [33] established the link between energy consumption and reliability. Energy-efficient parallel scheduling while meeting the real-time and reliability requirements has been studied well by proposing the shared recovery technique in uniprocessor systems [60]. The shared recovery technique means that a single recovery block can be shared among tasks to re-execute any faulty scaled task at runtime. Ref. [61] explored the three-stage (i.e., basic feasible schedule, task migration for high energy-efficiency, and reliability-aware slack exploitation) energy-efficient parallel scheduling using the checkpointing with rollback-recovery technique while simultaneously meeting the real-time and reliability requirements. However, the checkpointing with rollback-recovery technique is not a fault tolerant technique, such that its reliability enhancement ability is not sufficient.

Redundancy is an effective fault tolerant technique to meet the reliability requirement. Ref. [62] studied the energy-efficient parallel scheduling while meeting the reliability requirement using the dynamic redundancy technique in homogeneous systems. Ref. [63] addressed the energy-efficient fault-tolerant parallel scheduling while meeting the reliability requirement using the active replication technique (i.e., each task can be simultaneously replicated on several processors) in heterogeneous systems. In general, the functional safety requirement is the combination of real-time and reliability requirements; by using the active replication technique, Ref. [64] suggested the three-stage (i.e., response time verification, functional safety verification, and functional safety-critical energy optimization) energy-efficient parallel scheduling algorithm while simultaneously meeting the functional safety requirement.

(3) Energy-efficient parallel scheduling while meeting other requirements

Besides the real-time and reliability requirements, other requirements (e.g., cost budget and thermal requirements) have been studied recently. In Table 4, we survey the recent advances in heuristic energy-efficient parallel scheduling algorithms while meeting other requirements.

Ref. [65] provided the cost-to-budget technique to implement the energy-efficient parallel scheduling while meeting the cost budget requirement in heterogeneous systems; the cost-to-budget technique has two approaches of max-cost-down-to-budget and min-cost-up-to-budget. Ref. [66] developed the available budget preassignment technique to implement the energy-efficient parallel scheduling while meeting the cost budget requirement in heterogeneous systems; the available budget preassignment technique can shift the cost budget requirement of the parallel application to that of each task. Ref. [67] pursued a two-level (i.e., processor level and core level) technique to implement the energy-efficient parallel scheduling while meeting the real-time and thermal requirements in heterogeneous systems.

3.2 Meta-Heuristic Algorithms

Meta-heuristic algorithm is usually an improvement of the heuristic algorithm, and it is the product of the combination of the random algorithm and the local search algorithm. Meta-heuristic algorithm is an iterative generation process that uses heuristic algorithms to explore and develop the search space through intelligent combinations of different concepts. In this process, learning strategies are used to obtain and master information to effectively find approximate optimal solutions. Many meta-heuristic algorithms are inspired by some random phenomena in nature (e.g., genetic algorithm, simulated annealing algorithm).

TABLE 4
Recent Advances in Heuristic Energy-Efficient Parallel Scheduling Algorithms While Meeting Other Requirements

Reference	Year	Environment	Requirement	Technique	Static/dynamic
[65]	2018	Heterogeneous systems	Cost budget requirement	Available budget preassignment	Static
[66]	2018	Heterogeneous systems	Cost budget requirement	Cost-to-budget	Static
[67]	2018	Heterogeneous systems	Real-time requirement, thermal requirement	Processor level and core level	Static

TABLE 5
Recent Advances in Meta-Heuristic Energy-Efficient Parallel Scheduling Algorithms While Meeting the Real-Time Requirement

Reference	Year	Environment	Requirement	Technique	Description	Static/ dyanamic
[68]	2016	Heterogeneous systems	Real-time requirement	Evolutionary intelligence	Quantum-inspired hyper-heuristics	Static
[69]	2017	Heterogeneous systems	Real-time requirement	Evolutionary intelligence	Genetic algorithm	Static
[70]	2019	Heterogeneous systems	Real-time requirement	Evolutionary intelligence	Genetic algorithm	Static
[71]	2020	Heterogeneous systems	Real-time requirement	Evolutionary intelligence	Genetic algorithm	Static
[72]	2020	Heterogeneous systems	Real-time requirement	Evolutionary intelligence	Genetic algorithm	Static
[73]	2020	Heterogeneous systems	Real-time requirement	Evolutionary intelligence	Hybrid invasive weed optimization and culture	Static
[74]	2020	Heterogeneous systems	Real-time requirement	Biological intelligence	Cuckoo search algorithm	Static
[75]	2020	Heterogeneous systems	Real-time requirement	Swarm intelligence	Hybrid particle swarm optimization algorithm	Static

Meta-heuristic algorithms can be classified into three types, namely, evolutionary intelligence, biological intelligence, and swarm intelligence [12]. In Table 5, we survey the recent advances in meta-heuristic energy-efficient parallel scheduling algorithms while meeting the real-time requirement.

Ref. [68] introduced the hyper-heuristic framework, where the quantum-inspired hyper-heuristics technique is proposed to implement the energy-efficient parallel scheduling while meeting the real-time requirement in heterogeneous systems; the quantum-inspired hyper-heuristic technique belongs to the population-based evolutionary intelligence. Genetic algorithm is an evolutionary intelligence algorithm widely used in low-energy parallel scheduling, but it has limited search ability and unstable convergence rate. Refs. [69], [70], [71] improved genetic algorithms through accelerating and optimizing the evolution processes, such that they can be applied to energy-efficient parallel scheduling while meeting the real-time requirement in heterogeneous systems. The improved details are below: 1) Ref. [69] proposed an improved genetic algorithm by introducing the random evolution mechanism, specific crossover, and perturbation operations; 2) Ref. [70] described an improved genetic algorithm by introducing elite operators, elite mutations, and adaptive genetic optimization; and 3) Ref. [71] presented an improved genetic algorithm by introducing the spectrum partitioning algorithm. For deep learning-based parallel applications based on the Convolutional Neural Networks (CNN), Ref. [72] produced an improved genetic algorithm to implement the energy-efficient parallel scheduling while meeting the real-time requirement in heterogeneous systems.

Besides genetic algorithm, another evolutionary intelligence algorithm called invasive weed optimization has been used to implement the energy-efficient parallel scheduling while meeting the real-time requirement in heterogeneous systems. For instance, Ref. [73] improved the basic invasive weed optimization algorithm and proposed the hybrid invasive weed optimization and culture algorithm to address the problem of energy-efficient parallel scheduling while meeting the real-time requirement in heterogeneous systems.

Cuckoo search is a biological intelligence algorithm and has been studied in recent years. On the basis of Gaussian random walk and adaptive discovery probability, Ref. [74] showed an improved cuckoo search algorithm to achieve the energy-efficient parallel scheduling while meeting the real-time requirement in heterogeneous systems.

For swarm intelligence, Ref. [75] carried out the CPU-GPU utilization-aware energy-efficient parallel scheduling while meeting the real-time requirement in heterogeneous systems: 1) the first stage proposes the heuristic greedy strategy to solve the basic CPU-GPU utilization-aware energy-efficient scheduling problem; 2) the second stage improves the global optimization ability of the basic algorithm through proposing a novel hybrid particle swarm optimization algorithm.

3.3 Integer Programming Algorithms

At present, there are three types of integer programming algorithms, including integer linear programming, mixed integer linear programming, and non-linear mixed integer programming. Integer linear programming is the easiest and most used integer programming, and it refers specifically to the optimization problem where both the objective function and the constraints are linear. The problem of finding the maximum or minimum value of a linear objective function under linear constraints is called a linear programming problem. Linear programming is an important field in optimization problems. Many optimization problems can be decomposed into linear programming sub-problems, and we can solve these sub-problems one by one. Linear programming cannot obtain the optimal solution in polynomial time, but when the scale of the problem is not large, linear programming is a good way to solve the optimal solution.

The three types of algorithms (i.e., integer linear programming, mixed integer linear programming, and non-linear mixed integer programming) have currently been applied to

TABLE 6
Recent Advances in Integer Programming-Based Energy-Efficient Parallel Scheduling Algorithms While Meeting Various Requirements

Reference	Year	Environment	Requirement	Technique	Static/dynamic
[76]	2019	Heterogeneous systems	Real-time requirement	Integer linear programming	Static
[77]	2018	Heterogeneous systems	Real-time requirement, reliability requirement	Mixed integer linear programming	Static
[78]	2019	Heterogeneous systems	Real-time requirement	Non-linear mixed integer programming	Static

energy-efficient parallel scheduling. In Table 6, we survey the recent advances in integer programming-based energy-efficient parallel scheduling algorithms while meeting various requirements.

Ref. [76] first considered the task transition overhead in the energy-efficient parallel scheduling while meeting the real-time requirement in heterogeneous systems, where an integer linear programming algorithm based on task profile information was proposed; this algorithm extends integer linear programming formula and inserts transition instructions into the best program locations.

Ref. [77] applied a multi-stage approach to solve the energy-efficient parallel scheduling while meeting the real-time and reliability requirements in heterogeneous systems: 1) an energy-efficient scheduling scheme by fault tolerance is proposed to reduce the impact factors that affects reliability; 2) the optimal scheduling is obtained through the mixed integer linear programming algorithm; and 3) a binary particle swarm optimization based on list is proposed to deal the mapping and scheduling of application.

Ref. [78] used the non-linear mixed integer programming algorithm to conduct the energy-efficient parallel scheduling while meeting the real-time requirement in heterogeneous systems; the non-linear mixed integer programming algorithm is generally transferred to the mixed integer linear programming algorithm, and then the bound is added to find the optimal solution iteratively and recursively. The process in Ref. [78] consists of two stages: 1) finding the closed minimum energy consumption without considering response time; 2) optimizing response time by adjusting the task position and resource assignment.

3.4 Machine Learning Algorithms

Machine learning algorithm relies on historical collected data to achieve classification, prediction of system indicators; and such algorithm can be combined with other categories of scheduling algorithms to implement the energy-efficient parallel scheduling. Machine learning-based energy-efficient parallel scheduling algorithms are mostly suitable for dynamic application scenarios. However, most machine learning algorithms are designed for energy-efficient non-parallel scheduling algorithms [32], [81], [82], [83], [84], [85], [86], and only a few recent studies are designed for energy-efficient parallel scheduling algorithms [79], [80]. In Table 7, we survey the recent advances in machine learning-based energy-efficient parallel scheduling algorithms while meeting various requirements.

Reinforcement learning is popular for the energy-efficient parallel scheduling in recent years. Ref. [79] constructed a deep Q-learning model to learn the task request mode and the actual energy consumption, thereby implementing the energy-efficient parallel scheduling while meeting the user's service-level agreement requirement in heterogeneous systems. Ref. [80] formulated the reinforcement learning framework based on the Chebyshev scalar function to simultaneously achieve the energy-efficient parallel scheduling and reduce the response time while meeting the budget requirement in heterogeneous systems.

4 Energy-Aware Parallel Scheduling Algorithms

Through extensive search and analysis, the energy-aware parallel scheduling algorithms mainly involve three categories: 1) heuristic algorithms; 2) meta-heuristic algorithms; and 3) integer programming algorithms. Compared with the energy-efficient parallel scheduling, research works on the energy-aware parallel scheduling are relatively few, and most of them focus on the heuristic algorithm. For energy-aware parallel scheduling, there is only one work using the meta-heuristic algorithm [68] and only one work using the integer programming algorithm [93]; the game theory and machine learning algorithms are still blank. Energy-aware parallel scheduling is not suitable for game theory because it is a single objective optimization problem. As for machine learning, the main reason is that energy-aware parallel scheduling is still in its infancy compared to energy-efficient parallel scheduling and energy-conscious parallel scheduling. We believe that some energy-aware parallel scheduling algorithms using machine learning will come out in the near future. Fig. 7

TABLE 7
Recent Advances in Machine Learning-Based Energy-Efficient Parallel Scheduling Algorithms While Meeting Various Requirements

Year	Reference	Environment	Requirement	Technique	Static/dynamic
[79]	2018	Heterogeneous systems	Service-level agreement requirement	Reinforcement learning, Q-learning	Static
[80]	2019	Heterogeneous systems	Budget requirement	Reinforcement learning, Q-learning	Static

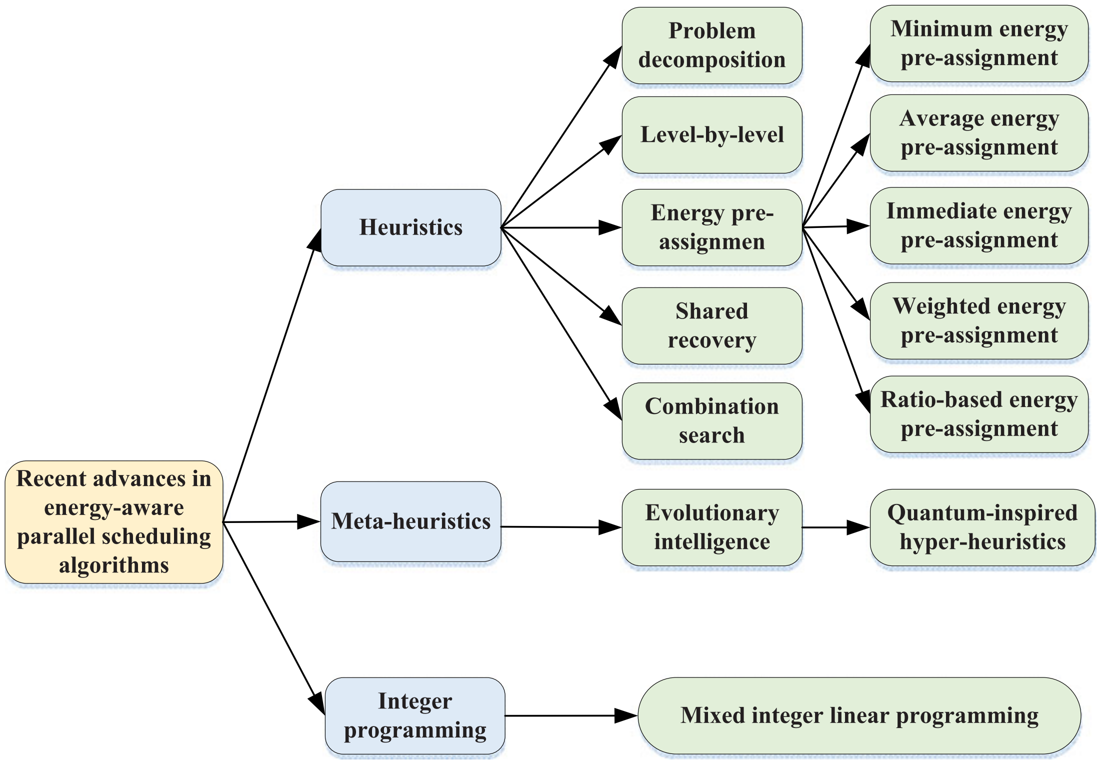

Recent advances in energy-aware parallel scheduling algorithms
Heuristics
Meta-heuristics
Integer programming
Problem decomposition
Level-by-level
Energy pre-assignmen
Shared recovery
Combination search
Minimum energy pre-assignment
Average energy pre-assignment
Immediate energy pre-assignment
Weighted energy pre-assignment
Ratio-based energy pre-assignment
Evolutionary intelligence
Quantum-inspired hyper-heuristics
Mixed integer linear programming

Fig. 7. Overall algorithms categories of recent advances in energy-aware parallel scheduling according to heuristics, meta-heuristics, and integer programming.

surveys the overall algorithm categories of recent advances in energy-aware parallel scheduling according to heuristics, meta-heuristics, and integer programming.

4.1 Heuristic Algorithms

(1) Energy-aware parallel scheduling towards response time reduction

In Table 8, we survey the recent advances in heuristic energy-aware parallel scheduling algorithms towards the response time reduction.

Besides the energy-efficient parallel scheduling, Refs. [43], [44], [45] studied the energy-aware parallel scheduling towards response time reduction in homogeneous systems using the same techniques; in other words, the three Refs. [43], [44], [45] simultaneously solved the dual problems of 1): energy-efficient parallel scheduling while meeting the real-time requirement and 2): energy-aware parallel scheduling towards response time reduction; in addition, the proposed technique of each study can be adapted to the dual problems. Ref. [87] considered the energy-aware parallel scheduling towards response time reduction in homogeneous systems and proposed the incremental static voltage adaptation technique; this technique constantly iteratively observes the deviation of the actual energy consumption exceeding the energy consumption constraint, and then adjusts the frequency of some key tasks until the actual energy consumption is within the given energy consumption constraint.

Energy-aware parallel scheduling towards response time reduction in heterogeneous systems has been addressed in 2016 [88], and an energy pre-assignment technique was first proposed; the core idea of the energy pre-assignment technique is to transfer the energy consumption constraint of the application to each task through pre-assigning certain energy consumptions to un-assigned tasks, thereby reducing the complexity of the problem. However, the energy pre-assignment technique in Ref. [88] is implemented through pre-assigning the minimum energy consumptions to un-assigned tasks (i.e., the minimum energy pre-assignment technique), resulting in a overly pessimistic energy pre-assignment and limited reduction of response time. Therefore, several subsequent works have made some improvements in energy pre-assignment techniques. Ref. [89] mentioned the average energy pre-assignment technique, which has lower pessimism than the average energy pre-assignment technique in Ref. [89]. Ref. [90] offered the immediate pre-assignment technique on the basis of the average pre-assignment technique, which has lower average response time than the average energy pre-assignment technique of Ref. [89]. Ref. [91] designed the weighted energy pre-assignment technique to improve the minimum energy pre-assignment technique proposed in Ref. [88]; however, our experimental comparison and analysis find that average response time obtained by the weighted pre-assignment technique is worse than those of obtained by the average pre-assignment technique in Ref. [89] and the immediate pre-assignment technique in Ref. [90]. By analyzing the characteristics and limitations of the previous energy pre-assignment techniques, Ref. [92] presented the ratio-based energy pre-assignment technique, which splits the available energy consumption in terms of the energy ratios of individual tasks with a customized manner; experimental results discover that average response time obtained by the ratio-based pre-assignment technique is better than those of obtained by the

TABLE 8
Recent Advances in Heuristic Energy-Aware Parallel Scheduling Algorithms Towards Response Time Reduction

Reference	Year	Environment	Objective	Technique	Static/dynamic
[43]	2012	Homogeneous systems	Response time reduction	Decomposing three subproblems	Static
[44]	2015	Homogeneous systems	Response time reduction	Level-by-level	Static
[45]	2016	Homogeneous systems	Response time reduction	List scheduling and list placement	Static
[87]	2009	Homogeneous systems	Response time reduction	Incremental static voltage adaptation	Static
[88]	2016	Heterogeneous systems	Response time reduction	Minimum energy pre-assignment	Static
[89]	2017	Heterogeneous systems	Response time reduction	Average energy pre-assignment	Static
[90]	2018	Heterogeneous systems	Response time reduction	Immediate energy pre-assignment	Static
[91]	2020	Heterogeneous systems	Response time reduction	Weighted energy pre-assignment	Static
[92]	2019	Heterogeneous systems	Response time reduction	Ratio-based energy pre-assignment	Static

TABLE 9
Recent Advances in Heuristic Energy-Aware Parallel Scheduling Algorithms Towards Reliability Enhancement (Some Works Also Meeting the Real-Time Requirement)

Reference	Year	Environment	Objective	Requirement	Technique	Static/dynamic
[94]	2015	Heterogeneous systems	Reliability enhancement	-	Finding optimal combination of reliability and energy	Static
[95]	2017	Heterogeneous systems	Reliability enhancement	-	Minimum energy pre-assignment	Static
[96]	2010	Uniprocessor systems	Reliability enhancement	Real-time requirement	Shared recovery	Dynamic
[97]	2019	Homogeneous systems	Reliability enhancement	Real-time requirement	Soft-error and lifetime enhancement simultaneously	Static
[98]	2018	Heterogeneous systems	Reliability enhancement	Real-time requirement	Average energy pre-assignment	Static

previous average pre-assignment techniques in Refs. [88], [89], [90].

(2) Energy-aware parallel scheduling towards reliability enhancement

Table 9 surveys the recent advances in heuristic energy-aware parallel scheduling algorithms towards reliability enhancement (some works also meet the real-time requirement).

Ref. [94] conducted the energy-aware parallel scheduling towards reliability enhancement in heterogeneous systems by finding an optimal combination of reliability and energy; however, this combination may not always meet the energy consumption constraint of the parallel application; the reason is that when a task is executed in a processor, its reliability and energy are monotonically increased. To overcome the above limitation, Ref. [95] discussed an energy-aware parallel scheduling algorithm towards reliability enhancement in heterogeneous systems through the minimum energy pre-assignment technique proposed in Ref. [88].

Some works also meeting the real-time requirement when conducting the energy-aware parallel scheduling towards reliability enhancement. Ref. [96] researched the energy-aware parallel scheduling towards reliability enhancement while meeting the real-time requirement in uniprocessor systems by using the same shared recovery technique as Ref. [60]. Ref. [97] considered the energy-aware parallel scheduling towards reliability (including soft-error and lifetime simultaneously) enhancement while meeting the real-time requirement in homogeneous systems. Ref. [98] adopted the average energy pre-assignment technique in Ref. [89] to implement the energy-aware parallel scheduling towards reliability enhancement while meeting the real-time requirement in heterogeneous systems.

4.2 Meta-Heuristic Algorithms

In Table 10, we survey the recent advance (only one work) in meta-heuristic energy-aware parallel scheduling algorithm towards response time reduction.

As mentioned in Section 3.2, Ref. [68] employed the quantum-inspired hyper-heuristic algorithm to implement the energy-efficient parallel scheduling; further, the energy-aware parallel scheduling can also be studied by using the same quantum-inspired hyper-heuristic algorithm. In other words, the quantum-inspired hyper-heuristic algorithm proposed in Ref. [68] can simultaneously solve the dual problems of energy-efficient parallel scheduling while meeting the real-time requirement and energy-aware parallel

TABLE 10
Recent Advance in Meta-Heuristic Energy-Aware Parallel Scheduling Algorithm Towards Response Time Reduction

Reference	Year	Environment	Objective	Technique	Static/dynamic
[68]	2016	Heterogeneous systems	Response time reduction	Quantum-inspired hyper-heuristics	static

TABLE 11
Recent Advance in Energy-Aware Parallel Scheduling Algorithm Using Integer Programming

Reference	Year	Environment	Objective	Technique	Static/dynamic
[93]	2019	Heterogeneous systems	Security requirement	Mixed-integer linear programming	Static

scheduling towards response time reduction in heterogeneous systems.

4.3 Integer Programming Algorithms

In Table 11, we survey the recent advance (only one work) in energy-aware parallel scheduling algorithm towards response time reduction using integer programming.

Ref. [93] produced the mixed integer linear programming algorithm to implement the energy-aware parallel scheduling towards security enhancement while meeting the real-time requirement in heterogeneous systems; to simultaneously solve the energy consumption constraint and security enhancement problems, the algorithm is implemented by adopting an analysis-based two-stage scheme: 1) energy-aware task assignment and frequency selection; 2) security-critical service selection.

5 Energy-Conscious Parallel Scheduling Algorithms

In this section, we survey recent advances in energy-conscious parallel scheduling algorithms. The problem of energy-conscious parallel scheduling is usually a bi-objective problem (i.e., Pareto optimal problem) or multi-objective optimization problem [99], [100]; therefore, integer liner programming that must have explicit constraints is no longer applicable; meanwhile, energy consumption reduction and response time reduction require contention for computing resources (processors); therefore, this contention can be considered as a game process, and game theory has been applied to energy-conscious parallel scheduling. Finally, the energy-conscious parallel scheduling algorithms mainly involve four categories: 1) heuristic algorithms; 2) meta-heuristic algorithms; 3) machine learning algorithms; and 4) game theory algorithms. Since integer programming restricts one or more constraints to the optimal solution of a certain objective, it is not suitable for energy-conscious parallel scheduling with multi-objective optimization. Fig. 8 surveys the overall algorithms categories of recent advances in energy-conscious parallel scheduling according to heuristics, meta-heuristics, machine learning, and game theory.

5.1 Heuristic Algorithms

Table 12 surveys the recent advances in heuristic energy-conscious parallel scheduling algorithms.

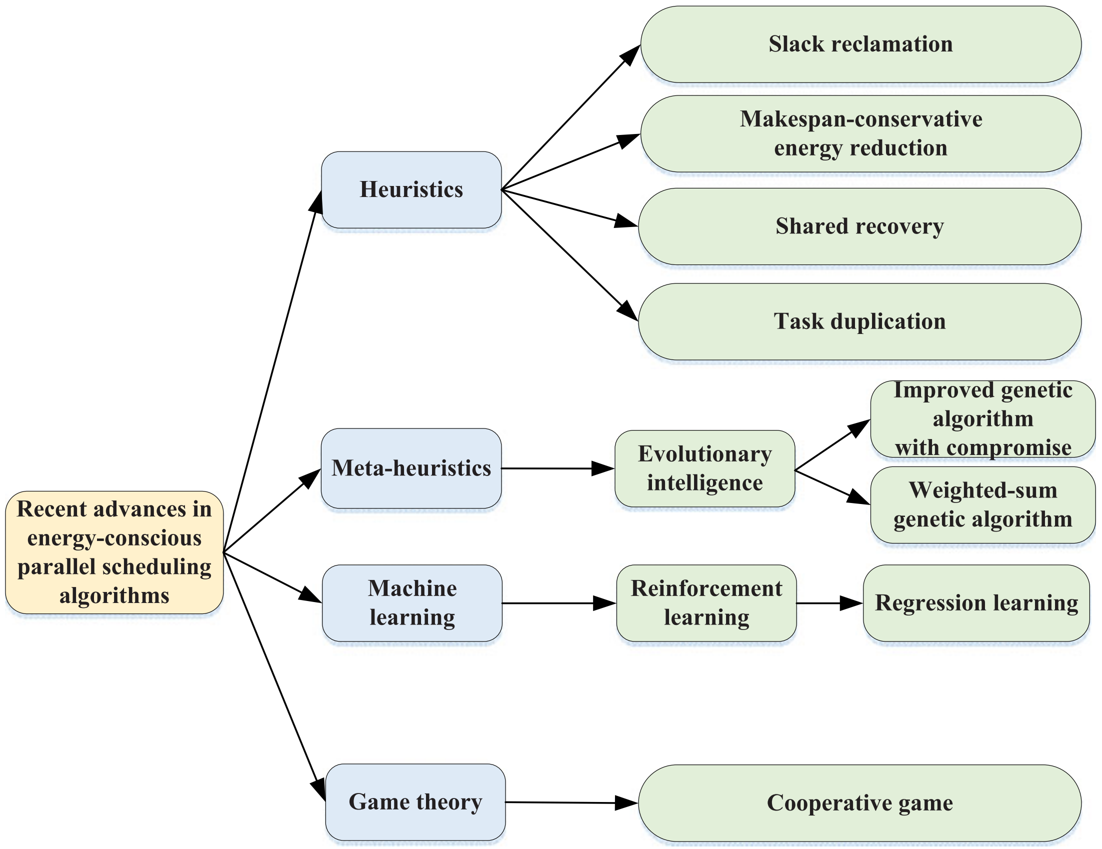

Recent advances in energy-conscious parallel scheduling algorithms
Heuristics
Slack reclamation
Makespan-conservative energy reduction
Shared recovery
Task duplication
Meta-heuristics
Evolutionary intelligence
Improved genetic algorithm with compromise
Weighted-sum genetic algorithm
Machine learning
Reinforcement learning
Regression learning
Game theory
Cooperative game

Fig. 8. Overall algorithms categories of recent advances in energy-conscious parallel scheduling according to heuristics, meta-heuristics, machine learning, and game theory.

TABLE 12
Recent Advances in Heuristic Energy-Conscious Parallel Scheduling Algorithms

Reference	Year	Environment	Objective	Technique	Static/dynamic
[99]	2011	Homogeneous systems	Response time reduction, energy consumption reduction	Task duplication and balanced task duplication	Static
[100]	2011	Heterogeneous systems	Response time reduction, energy consumption reduction	Makespan-conservative energy reduction	Static
[101]	2016	Heterogeneous systems	Reliability enhancement, energy consumption reduction	Shared recovery	Dynamic
[102]	2017	Heterogeneous systems	Response time reduction, energy consumption reduction	Slack reclamation	Static

TABLE 13
Recent Advances in Meta-Heuristic Energy-Conscious Parallel Scheduling Algorithms

Reference	Year	Environment	Objective	Technique	Static/dynamic
[104]	2017	Heterogeneous systems	Reliability enhancement, energy consumption reduction	Improved genetic algorithm with compromise	Static
[105]	2018	Homogeneous systems	Response time reduction, energy consumption reduction	Weighted-sum genetic algorithm	Static

TABLE 14
Recent Advances in Energy-Conscious Parallel Scheduling Algorithm Using Machine Learning

Reference	Year	Environment	Objective	Technique	Static/dynamic
[106]	2015	Heterogeneous systems	Response time reduction, energy consumption reduction	Regression learning	Static

Ref. [99] developed two energy-conscious parallel scheduling algorithms in homogeneous systems; the algorithms can simultaneously reduce the energy consumption and response time by using the task duplication and balanced task duplication techniques, respectively. The energy-conscious parallel scheduling algorithm in heterogeneous systems was first studied Ref. [100]; the algorithm implements the joint optimization of energy consumption reduction and response time reduction by using the objective function of relative superiority metric and proposing the makespan-conservative energy reduction technique. Through using the shared recovery technique described in Refs. [96], [103], the energy-conscious parallel scheduling algorithm that implements the joint optimization of energy consumption reduction and reliability enhancement in heterogeneous systems was designed in Ref. [101]. Based on the slack reclamation techniques proposed in Refs. [46], [48], [50], [51], five energy-conscious parallel scheduling algorithms (i.e., sequential, round robin, gap search, interleaving, and merge one) were devised in Ref. [102] to achieve the joint optimization of response time reduction and energy consumption reduction in heterogeneous systems.

5.2 Meta-Heuristic Algorithms

Similar to the meta-heuristic energy-efficient parallel scheduling algorithms shown in Section 3.2, some improved genetic algorithms are proposed to implement the energy-conscious parallel scheduling. In Table 13, we survey the recent advances in meta-heuristic energy-conscious parallel scheduling algorithms.

Ref. [104] presented an improved genetic algorithm with compromise to the energy-conscious parallel scheduling, which achieves the joint optimization of reliability enhancement and energy consumption reduction in heterogeneous systems. Ref. [105] put forward the weighted-sum genetic algorithm to achieve the energy-conscious parallel scheduling algorithm, which implements the joint optimization of response time reduction and energy consumption reduction in homogeneous systems.

5.3 Machine Learning Algorithms

Ref. [106] investigated the adaptive energy-conscious parallel scheduling by using the regression learning technique to implement the trade off between response time reduction and energy consumption reduction in heterogeneous systems. In Table 14, we survey such recent energy-conscious parallel scheduling algorithm.

5.4 Game Theory Algorithms

As emphasized in the preface of Section 5, response time reduction and energy consumption reduction are achieved by competing for processor resources, such that the contention for processor resources is a game process. Non-cooperative game and cooperative game are two main games for scheduling. Most game theory algorithms are designed for energy-conscious non-parallel scheduling algorithms [109], [110],

TABLE 15
Recent Advances in Energy-Conscious Parallel Scheduling Algorithms Using Game Theory

Reference	Year	Environment	Objective	Technique	Static/dynamic
[107]	2008	Heterogeneous systems	Response time reduction, energy consumption reduction	Cooperative game	Static
[108]	2013	Heterogeneous systems	Response time reduction, energy consumption reduction	Cooperative game	Static

[111], [112], while only a few recent studies are proposed for energy-conscious parallel scheduling [107], [108]. Due to the data dependency in a parallel application, cooperative game is used for energy-conscious parallel scheduling. In Table 15, we survey the recent advances in energy-conscious parallel scheduling algorithms using game theory.

Ref. [107] studied the energy-conscious parallel scheduling algorithm by using the Nash bargaining solution in heterogeneous systems; the bargaining point was determined by applying the classic game theory techniques of Kuhn-Tucker condition and Lagrangian. Ref. [108] considered the game theory-based energy-conscious parallel pre-scheduling and scheduling algorithm in homogeneous systems; this algorithm uses Nash equilibrium to determine the most appropriate task merging to jointly optimization the response time reduction and energy consumption reduction.

6 Future Trends

6.1 New Requirements

From the recent advances in Sections 3, 4, and 5, we can see that many researchers have conducted a lot of studies in various low-energy parallel scheduling algorithms; however, low-energy parallel scheduling algorithms are facing the following new requirements.

(1) *Energy-efficient parallel scheduling algorithms should consider the joint requirement of safety and security.* With the continuous development of intelligence and networking, computing systems inevitably face increasingly serious security issues. In the past, only functional safety requirement is considered in energy-efficient parallel scheduling algorithms for sustainable computing systems, but now both the functional safety and cyber security requirements should be considered together. Therefore, proposing energy-efficient parallel scheduling algorithm while simultaneously meeting functional safety and cyber security requirements is a main current challenge.

(2) *Energy-aware parallel scheduling algorithms should consider the joint constraint of dynamic, static and network energy consumption.* With the development of 5G, some sustainable computing systems (i.e., IoT) introduces a large number of intelligent devices and increases the amount of data transmission. Decreasing the network energy consumption generated by huge data transmission is a challenge. At the same time, most computing devices are powered by batteries, and meeting system energy consumption (including dynamic and static energy consumptions) constraint is a main research topic; however, most related works ignore the static energy consumption because it is unmanageable. With the increased deployment of system softwares and middlewares in IoT devices, the proportion of static energy consumption continues to increase in the proportion of total energy consumption [18]. Proposing energy-aware parallel scheduling algorithm while meeting dynamic, static and network energy consumption constraints is one of the main current challenges.

(3) *Energy-conscious parallel scheduling algorithms face the conflicting pursuits of service providers and users.* For instance, service providers and users are two main roles in CCS, but they have conflicting pursuits [113]. High performance (e.g., high throughout) and low resource consumption (including low energy consumption, low task redundancy, etc.) is the core pursuit of service providers, while high QoS metrics (e.g., real-time, reliability, security, cost, etc.) are the main pursuit of users [113]. jointly improving the individual pursuits of service providers and users is a conflicting problem. With the increasing complexity of sustainable computing systems, solving the above conflicting problem is a long-term challenge.

In view of the above challenges, the future low-energy parallel scheduling are challenged by multiple factors, such as 1) joint requirements of safety and security; 2) joint constraints of dynamic, static and network energy consumption; and 3) conflicting pursuits of service providers and users. These challenges mean that the current research for low-energy parallel scheduling algorithms is not yet complete. It is necessary to survey recent advances in low-energy parallel scheduling algorithms according to different scheduling styles, thereby providing a core foundation for solving the above challenges.

6.2 Future Developments

Currently, sustainable computing systems are continuously integrating complex hardware and intelligent functions, and are developing to be merged together into a common complex large-scale system architecture, namely, end-edge-cloud orchestrated architecture [114]. In this end-edge-cloud orchestrated architecture, we need to comprehensively consider the following three future developments for low-energy parallel scheduling algorithms.

(1) Application Scenarios. End-edge-cloud orchestrated architecture makes the development of complex systems easier, but it causes the system itself more complex. Therefore, low-energy parallel scheduling algorithms will be increasingly tied to specific application scenarios. If the application scenarios are different, the application model will be adjusted accordingly, and the corresponding low-energy parallel scheduling algorithms will change. Therefore, the future low-energy parallel scheduling algorithm will be deeply integrated with the application scenarios, and the algorithm design will be customized according to the specific application scenario. For example, a smart factory is a typical end-edge-cloud orchestrated architecture, and it includes three levels: hardware level (end), development level (edge), and an operation level (cloud). These

levels have different functional requirements, such that the low-energy parallel scheduling algorithm design for each level will be customized according to the characteristics of the application scenario of that level.

(2) Hardware Features. At present, commercial processors not only have DVFS processors that support continuous frequency adjustment, but also have many other DVFS processors that support discrete frequency adjustment, and even have processors that do not support DVFS technology. DVFS processors with different hardware features have different effects on the complexity of low-energy parallel scheduling, and the corresponding algorithm design will change. Therefore, the low-energy parallel scheduling algorithm will be deeply combined with complex hardware environments (e.g., the integration of CPU, GPU, and DSP). Future low-energy parallel scheduling algorithms will be customized according to the hardware features.

(3) Algorithm Combination. In addition to customizing low-energy parallel scheduling algorithms based on application scenarios and hardware features, the choice of algorithm type is also important. Current low-energy parallel scheduling algorithms can not completely meet the future diverse and complicated requirements. At present, Low-energy parallel scheduling algorithms such as heuristics, meta-heuristics, integer programming, game theory, and machine learning have basically reached the bottleneck. To further break through the technical bottleneck, future research will use a combination of advanced algorithms, such as clustering, game theory, reinforcement learning. Moreover, we need to feedback energy optimization efficiency and improve system resource assignment in the way of theoretical and practical co-verification.

7 Conclusion

This paper surveys recent advances and future trends in low-energy parallel scheduling algorithms according to multiple computing system domains with individual scheduling styles, namely, 1) energy-efficient parallel scheduling algorithms; 2) energy-aware parallel scheduling algorithms; and 3) energy-conscious parallel scheduling algorithms. The reviewed algorithms involve five categories of 1) heuristics; 2) meta-heuristics; 3) integer programming; 4) machine learning; and 5) game theory. Finally, we discuss the new requirements and future developments in low-energy parallel scheduling algorithms; new algorithms will be directly oriented to the end-edge-cloud orchestrated architecture, into which computing systems are developing into an integration.

Energy-efficient parallel scheduling algorithms reduce the energy consumption with an efficient manner while meeting one or multiple requirements. Energy-aware parallel scheduling algorithms improve the performance as much as possible while meeting the energy consumption constraint. Energy-conscious parallel scheduling algorithms simultaneously optimize energy consumption and other multiple performances metrics. The above three styles of scheduling algorithms collectively construct recent advances in low-energy parallel scheduling algorithms.

Acknowledgments

The authors would like to thank the associate editor and three anonymous reviewers for their constructive comments which have helped to improve the quality of the paper. This work was supported in part by the National Natural Science Foundation of China under Grants 61972139, 61932010, 61702172, 61672217, and 61872135, the Open Research Project of the Electronic Information and Control of Fujian University Engineering Research Center, Minjiang University, China under GrantMJXY-KF-EIC1902, and the Fundamental Research Funds for the Central Universities, Hunan University, China.

References

[1] H. Hagedoorn, “AMD Ryzen 9 3950X review - power consumption and temperatures,” 2019. [Online]. Available: https://www.guru3d.com/articles-pages/amd-ryzen-9–3950x-review,7.html
[2] F. Leferink and C. Keyer, “Electromagnetic interference in smart grids,” *Int. Workshop Energy*, Open Univ. Twente, Enschede, Netherlands, p. 2, May, 2017.
[3] M. Irfan, J. Abbas, S. M. Shaheed, and M. S. Nadeem, “A conspicuous survey of green computing environmental impact,” *Int. J. Innov. Appl. Stud.*, vol. 13, no. 4, pp. 935–945, Apr. 2015.
[4] P. Delforge, “America’s data centers consuming and wasting growing amounts of energy,” 2015. [Online]. Available: https://www.nrdc.org/resources/americas-data-centers-consuming-and-wasting-growing-amounts -energy
[5] Y. W. Zhang, H. Z. Zhang, and C. Wang, “Reliability-aware low energy scheduling in real time systems with shared resources,” *Microprocessors Microsyst.*, vol. 52, pp. 312–324, Jul. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S01419331163028l2
[6] S. Li and F. Broekaert, “Low-power scheduling with DVFS for common RTOS on multicore platforms,” *ACM SIGBED Rev.*, vol. 11, pp. 32–37, Feb. 2014.
[7] Enhanced Intel SpeedStep technology for the Intel pentium M processor, 2014. [Online]. Available: http://download.intel.com/design/network/papers/30117401.pdf
[8] AMD PowerNow! Technology informational white paper, 2000. [Online]. Available: www.amd-k6.com/wpbcontent/uploads/2012/07/24404a.pdf
[9] K. Flautner and D. Flynn, “A combined hardware-software approach for low-power SoCs: Applying adaptive voltage scaling and intelligent energy management software,” in *Prof. DesignCon Conf.*, 2003.
[10] NSF 18-538, “Cyber-physical systems (CPS),” *NSF Special Publication*, 2018. [Online]. Available: https://www.nsf.gov/pubs/2018/nsf18538/nsf18538.htm
[11] R. Bhattacharyya, A. Das, A. Majumdar, and P. Ghosh, “Real-Time Scheduling Approach for IoT-Based Home Automation System,” in *Proc. Data Manage. Anal. Innov.*, 2019, pp. 103–113.
[12] M. J. Usman *et al.*, “Energy-efficient nature-inspired techniques in cloud computing datacenters,” *Telecommun. Syst.*, vol. 71, no. 2, pp. 275–302, 2019.
[13] C. Schmittner, Z. Ma, E. Schoitsch, and T. Gruber, “A case study of FMVEA and CHASSIS as safety and security co-analysis method for automotive cyber-physical systems,” in *Proc. 1st ACM Workshop Cyber-Phys. Syst. Secur.*, 2015, pp. 69–80.
[14] B. Yu, L. Pu, Q. Xie, and J. Xu, “Energy efficient scheduling for IoT applications with offloading, user association and BS sleeping in ultra dense networks,” in *Proc. 16th Int. Symp. Model. Optim. Mobile Ad Hoc Wireless Netw.*, 2018, pp. 1–6.
[15] X. Lyu *et al.*, “Optimal schedule of mobile edge computing for Internet of Things using partial information,” *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2606–2615, Nov. 2017.
[16] W. Chen, G. Xie, R. Li, Y. Bai, C. Fan, and K. Li, “Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems,” *Future Gener. Comput. Syst.*, vol. 74, pp. 1–11, Sep. 2017.
[17] N. B. M. Nor, M. B. Hussin, and H. BinSelamat, “Energy Management for cloud computing: A survey from scheduling perspective of heuristic, game theory and learning strategy,” *J. Comput. Sci. Comput. Math.*, vol. 4, no. 3, pp. 41–48, Mar. 2014.

[18] M. Bambagini, M. Marinoni, H. Aydin, and G. Buttazzo, "Energy-aware scheduling for real-time systems: A survey," *ACM Trans. Embedded Comput. Syst.*, vol. 15, 2016, Art. no. 7.

[19] H. F. Sheikh and I. Ahmad, "Sixteen heuristics for joint optimization of performance, energy, and temperature in allocating tasks to multi-cores," *ACM Trans. Parallel Comput.*, vol. 3, 2016, Art. no. 9.

[20] S. Pagani, P. D. Sai Manoj, A. Jantsch, and J. Henkel, "Machine learning for power, energy, and thermal management on multicore processors: A survey," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 1, pp. 101–116, Jan. 2020.

[21] M. Qiu, L. T. Yang, Z. Shao, and E. H.-M. Sha, "Dynamic and leakage energy minimization with soft real-time loop scheduling and voltage assignment," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 3, pp. 501–504, Mar. 2010.

[22] J. Li, M. Qiu, J. W. Niu, L. T. Yang, Y. Zhu, and Z. Ming, "Thermal-aware task scheduling in 3D chip multiprocessor with real-time constrained workloads," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 2, pp. 1–22, Feb. 2013.

[23] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 682–694, Mar. 2014.

[24] M. Naghibzadeh, "Modeling and scheduling hybrid workflows of tasks and task interaction graphs on the cloud," *Future Gener. Comput. Syst.*, vol. 65, pp. 33–45, 2016.

[25] S. K. Roy, R. Devaraj, A. Sarkar, K. Maji, and S. Sinha, "Contention-aware optimal scheduling of real-time precedence-constrained task graphs on heterogeneous distributed systems," *J. Syst. Archit.*, vol. 105, May 2020, Art. no. 101706.

[26] R. Wu, "Dynamic scheduling strategy for block parallel cholesky factorization based on activity on edge network," *IEEE Access*, vol. 7, pp. 66317–66324, May 2019.

[27] S. Wunderlich, J. A. Cabrera, F. H. P. Fitzek, and M. Reisslein, "Network coding in heterogeneous multicore IoT nodes with DAG scheduling of parallel matrix block operations," *IEEE Internet of Things J.*, vol. 4, no. 4, pp. 917–933, Aug. 2017.

[28] M. Adhikari and S. Koley, "Cloud computing: A multi-workflow scheduling algorithm with dynamic reusability," *Arabian J. Sci. Eng.*, vol. 43, no. 2, pp. 645–660, Feb. 2018.

[29] M. Adhikari, T. Amgoth, and S. N. Srirama, "A survey on scheduling strategies for workflows in cloud environment and emerging trends," *ACM Comput. Surv.*, vol. 52, no. 4, Apr. 2019, Art. no. 68.

[30] L. Zhang, D. Zhao, and J. He, "Research of Distributed Vehicle Electronic and Electrical Architecture,"in *Proc. SAE-China Congr.: Sel. Papers* 2015, pp. 455–462.

[31] T. Mitra, "Heterogeneous multi-core architectures," *Inf. Media Technol.*, vol. 10, no. 3, pp. 383–394, Sep. 2015.

[32] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, and P. Li, "Energy-efficient scheduling for real-time systems based on deep Q-learning model," *IEEE Trans. Sustain. Comput.*, vol. 4, no. 1, pp. 132–141, First Quarter 2019.

[33] D. Zhu, R. Melhem, and D. Mossé, "The effects of energy management on reliability in real-time embedded systems," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Des.*, 2004, pp. 35–40.

[34] K. Li, "Power and performance management for parallel computations in clouds and data centers," *J. Comput. Syst. Sci.*, vol. 82, pp. 174–190, 2016.

[35] K. Li, "Energy and time constrained task scheduling on multiprocessor computers with discrete speed levels," *J. Parallel Distrib. Comput.*, vol. 95, pp. 15–28, 2016.

[36] K. Li, "Energy constrained scheduling of stochastic tasks," *J. Supercomput.*, vol. 74, pp. 485–508, 2018.

[37] R. Melhem, D. Mosse, and E. Elnozahy, "The interplay of power management and fault recovery in real-time systems," *IEEE Trans. Comput.*, vol. 53, no. 2, pp. 217–231, Feb. 2004.

[38] T. Wei, P. Mishra, K. Wu, and H. Liang, "Fixed-priority allocation and scheduling for energy-efficient fault tolerance in hard real-time multiprocessor systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 11, pp. 1511–1526, Nov. 2008.

[39] M. A. Haque, H. Aydin, and D. Zhu, "Energy-aware standby-sparing technique for periodic real-time applications," in *Proc. IEEE 29th Int. Conf. Comput. Des.*, 2011, pp. 190–197.

[40] S. Pagani and J.-J. Chen, "Energy efficient task partitioning based on the single frequency approximation scheme," in *Proc. IEEE 34th Real-Time Syst. Symp.*, 2013, pp. 308–318.

[41] Y. Qin, G. Zeng, R. Kurachi, Y. Matsubara, and H. Takada, "Execution-variance-aware task allocation for energy minimization on the big.little architecture," *Sustain. Comput., Informat. Syst.*, vol. 22, pp. 155–166, 2019.

[42] A. Colin, A. Kandhalu, and R. Rajkumar, "Energy-efficient allocation of real-time applications onto single-ISA heterogeneous multicore processors," *J. Signal Process. Syst.*, vol. 84, pp. 91–110, 2016.

[43] K. Li, "Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers," *IEEE Trans. Comput.*, vol. 61, no. 12, pp. 1668–1681, Dec. 2012.

[44] K. Li, "Power and performance management for parallel computations in clouds and data centers," *J. Comput. Syst. Sci.*, vol. 82, pp. 174–190, 2016.

[45] K. Li, "Energy and time constrained task scheduling on multiprocessor computers with discrete speed levels," *J. Parallel Distrib. Comput.*, vol. 95, pp. 15–28, 2016.

[46] Q. Huang, S. Su, J. Li, P. Xu, K. Shuang, and X. Huang, "Enhanced energy-efficient scheduling for parallel applications in cloud," in *Proc. 12th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, 2012, pp. 781–786.

[47] I. Pietri and R. Sakellariou, "Energy-aware workflow scheduling using frequency scaling," in *Proc. 43rd Int. Conf. Parallel Process. Workshops*, 2014, pp. 104–113.

[48] S. Su, Q. Huang, J. Li, X. Cheng, P. Xu, and K. Shuang, "Enhanced energy-efficient scheduling for parallel tasks using partial optimal slacking," *Comput. J.*, vol. 58, pp. 246–257, 2015.

[49] Y. Hu, C. Liu, K. Li, X. Chen, and K. Li, "Slack allocation algorithm for energy minimization in cluster systems," *Future Gener. Comput. Syst.*, vol. 74, pp. 119–131, Sep. 2017.

[50] G. Xie, J. Jiang, Y. Liu, R. Li, and K. Li, "Minimizing energy consumption of real-time parallel applications using downward and upward approaches on heterogeneous systems," *IEEE Trans. Ind. Informat.*, vol. 13, no. 3, pp. 1068–1078, Jun. 2017.

[51] G. Xie, G. Zeng, X. Xiao, R. Li, and K. Li, "Energy-efficient scheduling algorithms for real-time parallel applications on heterogeneous distributed embedded systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 12, pp. 3426–3442, Dec. 2017.

[52] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment," *J. Grid Comput.*, vol. 14, pp. 55–74, 2016.

[53] G. Xie, G. Zeng, R. Li, and K. Li, "Energy-aware processor merging algorithms for deadline constrained parallel applications in heterogeneous cloud computing," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 2, pp. 62–75, Second Quarter 2017.

[54] G. Xie, G. Zeng, J. Jiang, C. Fan, R. Li, and K. Li, "Energy management for multiple real-time workflows on cyber–physical cloud systems," *Future Gener. Comput. Syst.*, vol. 105, pp. 916–931, 2020.

[55] N. Gao, C. Xu, X. Peng, H. Luo, W. Wu, and G. Xie, "Energy-efficient scheduling optimization for parallel applications on heterogeneous distributed systems," *J. Circuits Syst. Comput.*, vol. 29, no. 13, p. 2050203, Oct. 2020, doi: 10.1142/S0218126620502035.

[56] M. Safari and R. Khorsand, "Energy-aware scheduling algorithm for time-constrained workflow tasks in DVFS-enabled cloud environment," *Simul. Modelling Practice Theory*, vol. 87, pp. 311–326, 2018.

[57] K. Li, "Optimal task execution speed setting and lower bound for delay and energy minimization," *J. Parallel Distrib. Comput.*, vol. 123, pp. 13–25, 2019.

[58] X. Tang, W. Shi, and F. Wu, "Interconnection network energy-aware workflow scheduling algorithm on heterogeneous systems," *IEEE Trans. Ind. Informat.*, vol. 16, no. 12, pp. 7637–7645, Dec. 2020.

[59] X. Li, W. Yu, R. Ruiz, and J. Zhu, "Energy-aware cloud workflow applications scheduling with geo-distributed data," *IEEE Trans. Services Comput.*, early access, Jan. 09, 2020, doi: 10.1109/TSC.2020.2965106.

[60] B. Zhao, H. Aydin, and D. Zhu, "Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, pp. 99–109, 2013.

[61] T. Wu, H. Gu, J. Zhou, T. Wei, X. Liu, and M. Chen, "Soft error-aware energy-efficient task scheduling for workflow applications in DVFS-enabled cloud," *J. Syst. Archit.*, vol. 84, pp. 12–27, 2018.

[62] M. Salehi *et al.*, "DRVS: Power-efficient reliability management through dynamic redundancy and voltage scaling under variations," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Des.*, 2015, pp. 225–230.

[63] G. Xie, Y. Chen, X. Xiao, Y. Chen, R. Li, and K. Li, "Energy-efficient fault-tolerant scheduling of reliable parallel applications on heterogeneous distributed embedded systems," *IEEE Trans. Sustain. Comput.*, vol. 3, no. 3, pp. 167–181, Jul.–Sep. 2018.

[64] G. Xie, H. Peng, J. Huang, R. Li, and K. Li, "Energy-efficient functional safety design methodology using ASIL decomposition for automotive cyber-physical systems," early access, Jun. 03, 2019, doi: 10.1109/TR.2019.2915818.

[65] J. Junqiang, W. Li, L. Pan, B. Yang, and X. Peng, "Energy optimization heuristics for budget-constrained workflow in heterogeneous computing system," *J. Circuits Syst. Comput.*, vol. 28, p. 1950159, Sep. 2018.

[66] Y. Chen, G. Xie, and R. Li, "Reducing energy consumption with cost budget using available budget preassignment in heterogeneous cloud computing systems," *IEEE Access*, vol. 6, pp. 20 572–20 583, 2018.

[67] J. Zhou *et al*., "Thermal-aware correlated two-level scheduling of real-time tasks with reduced processor energy on heterogeneous MPSoCs," *J. Syst. Archit.*, vol. 82, pp. 1–11, 2018.

[68] S. Chen, Z. Li, B. Yang, and G. Rudolph, "Quantum-inspired hyper-heuristics for energy-aware scheduling on heterogeneous computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 6, pp. 1796–1810, Jun. 2016.

[69] A. Mahmood, S. Khan, F. Albalooshi, and N. Awwad, "Energy-aware real-time task scheduling in multiprocessor systems using a hybrid genetic algorithm," *Electronics*, vol. 6, 2017, Art. no. 40.

[70] Y. Yun, E. J. Hwang, and Y. H. Kim, "Adaptive genetic algorithm for energy-efficient task scheduling on asymmetric multiprocessor system-on-chip," *Microprocess Microsyst.*, vol. 66, pp. 19–30, 2019.

[71] G. Taheri, A. Khonsari, R. Entezari-Maleki, and L. Sousa, "A hybrid algorithm for task scheduling on heterogeneous multiprocessor embedded systems," *Appl. Soft Comput.*, vol. 93, pp. 106–202, 2020.

[72] D. Kang, J. Oh, J. Choi, Y. Yi, and S. Ha, "Scheduling of deep learning applications onto heterogeneous processors in an embedded device," *IEEE Access*, vol. 8, pp. 43 980–43 991, 2020.

[73] P. Hosseinioun, M. Kheirabadi, S. R. K. Tabbakh, and R. Ghaemi, "A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm," *J. Parallel Distrib. Comput.*, vol. 143, pp. 88–96, 2020.

[74] Z. Deng, Z. Yan, H. Huang, and H. Shen, "Energy-aware task scheduling on heterogeneous computing systems with time constraint," *IEEE Access*, vol. 8, pp. 23 936–23 950, 2020.

[75] X. Tang and Z. Fu, "CPU–GPU utilization aware energy-efficient scheduling algorithm on heterogeneous computing systems," *IEEE Access*, vol. 8, pp. 58 948–58 958, 2020.

[76] Y. Qin, G. Zeng, R. Kurachi, Y. Li, Y. Matsubara, and H. Takada, "Energy-efficient intra-task DVFS scheduling using linear programming formulation," *IEEE Access*, vol. 7, pp. 30 536–30 547, 2019.

[77] K. Huang *et al*., "Energy-efficient fault-tolerant mapping and scheduling on heterogeneous multiprocessor real-time systems," *IEEE Access*, vol. 6, pp. 57 614–57 630, 2018.

[78] B. Hu, Z. Cao, and M. Zhou, "Scheduling real-time parallel applications in cloud to minimize energy consumption," *IEEE Trans. Cloud Comput.*, early access, Nov. 28, 2019, doi: 10.1109/TCC.2019.2956498.

[79] M. Cheng, J. Li, and S. Nazarian, "DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," in *Proc. 23rd Asia South Pacific Des. Autom. Conf.*, 2018, pp. 129–134.

[80] Y. Qin, H. Wang, S. Yi, X. Li, and L. Zhai, "An energy-aware scheduling algorithm for budget-constrained scientific workflows based on multi-objective reinforcement learning," *J. Supercomput.*, vol. 76, pp. 455–480, 2020.

[81] F. Farahnakian, P. Liljeberg, and J. Plosila, "Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning," in *Proc. 22nd Euromicro Int. Conf. Parallel Distrib. Netw.-Based Process.*, 2014, pp. 500–507.

[82] L. Lei, T. X. Vu, L. You, S. Fowler, and D. Yuan, "Efficient minimum-energy scheduling with machine-learning based predictions for multiuser MISO systems," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–6.

[83] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, S. U. Khan, and P. Li, "A double deep Q-learning model for energy-efficient edge scheduling," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 739–749, Sep./Oct. 2019.

[84] H. Huang, M. Lin, L. T. Yang, and Q. Zhang, "Autonomous power management with double-Q reinforcement learning method," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 1938–1946, Mar. 2020.

[85] S. Bourhnane, M. Abid, and R. Lghoul, "Machine learning for energy consumption prediction and scheduling in smart buildings," *SN Appl. Sci.*, vol. 2, 2020, Art. no. 297.

[86] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, "Q-learning based dynamic task scheduling for energy-efficient cloud computing," *Future Gener. Comput. Syst.*, vol. 108, pp. 361–371, 2020.

[87] I. Ahmad, R. Arora, D. White, V. Metsis, and R. Ingram, "Energy-constrained scheduling of DAGs on multi-core processors," *Proc. Int. Conf. Contemp. Comput.*, 2009, pp. 592–603.

[88] X. Xiao, G. Xie, R. Li, and K. Li, "Minimizing schedule length of energy consumption constrained parallel applications on heterogeneous distributed systems," in *Proc. 14th IEEE Int. Symp. Parallel Distrib. Process. Appl.*, 2016, pp. 1471–1476.

[89] J. Song, G. Xie, R. Li, and X. Chen, "An efficient scheduling algorithm for energy consumption constrained parallel applications on heterogeneous distributed systems," in *Proc. 15th IEEE Int. Symp. Parallel Distrib. Process. Appl.*, 2017, pp. 32–39.

[90] J. Li, G. Xie, K. Li, and Z. Tang, "Enhanced parallel application scheduling algorithm with energy consumption constraint in heterogeneous distributed systems," *J. Circuits Syst. Comput.*, vol. 28, pp. 1 950 190–1–1 950 190-23, Jan. 2019.

[91] Z. Quan, Z.-J. Wang, T. Ye, and S. Guo, "Task scheduling for energy consumption constrained parallel applications on heterogeneous computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 5, pp. 1165–1182, May 2020.

[92] G. Xie, J. Huang, Y. L. R. Li, and K. Li, "System-level energy-aware design methodology towards end-to-end response time optimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, Jun. 06, 2019, doi: 10.1109/TCAD.2019.2921350.

[93] J. Zhou *et al*., "Security-critical energy-aware task scheduling for heterogeneous real-time MPSoCs in IoT," *IEEE Trans. Services Comput.*, vol. 13, no. 4, pp. 745–758, Jul./Aug. 2020.

[94] L. Zhang, K. Li, Y. Xu, J. Mei, F. Zhang, and K. Li, "Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster," *Inf. Sci.*, vol. 319, pp. 113–131, 2015.

[95] X. Xiao, G. Xie, C. Xu, C. Fan, R. Li, and K. Li, "Maximizing reliability of energy constrained parallel applications on heterogeneous distributed systems," *J. Comput. Sci.*, vol. 26, pp. 344–353, 2017.

[96] B. Zhao, H. Aydin, and D. Zhu, "On maximizing reliability of real-time embedded applications under hard energy constraint," *IEEE Trans. Ind. Informat.*, vol. 6, no. 3, pp. 316–328, Aug. 2010.

[97] J. Zhou *et al*., "Resource management for improving soft-error and lifetime reliability of real-time MPSoCs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 12, pp. 2215–2228, Dec. 2019.

[98] G. Xie *et al*., "Reliability enhancement towards functional safety goal assurance in energy-aware automotive cyber-physical systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 12, pp. 5447–5462, Dec. 2018.

[99] Z. Zong, A. Manzanares, X. Ruan, and X. Qin, "EAD and PEBD: Two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters," *IEEE Trans. Comput.*, vol. 60, no. 3, pp. 360–374, Mar. 2011.

[100] Y. C. Lee and A. Y. Zomaya, "Energy conscious scheduling for distributed computing systems under different operating conditions," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 8, pp. 1374–1381, Aug. 2011.

[101] L. Zhang, K. Li, K. Li, and Y. Xu, "Joint optimization of energy efficiency and system reliability for precedence constrained tasks in heterogeneous systems," *Int. J. Elect. Power Energy Syst.*, vol. 78, pp. 499–512, 2016.

[102] J. Junqiang, Y. Lin, G. Xie, L. Fu, and J. Yang, "Time and energy optimization algorithms for the static scheduling of multiple workflows in heterogeneous computing system," *J. Grid Comput.*, vol. 15, pp. 435–456, Feb. 2017.

[103] L. Zhao, Y. Ren, and K. Sakurai, "Reliable workflow scheduling with less resource redundancy," *Parallel Comput.*, vol. 39, pp. 567–585, 2013.

[104] L. Zhang, K. Li, C. Li, and K. Li, "Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems," *Inf. Sci.*, vol. 379, pp. 241–256, 2017.

[105] A. S. Pillai, K. Singh, V. Saravanan, A. Anpalagan, I. Woungang, and L. Barolli, "A genetic algorithm-based method for optimizing the energy consumption and performance of multiprocessor systems," *Soft Comput.*, vol. 22, no. 10, pp. 3271–3285, Oct. 2018.

[106] S. Yang *et al.*, "Adaptive energy minimization of embedded heterogeneous system using regression-based learning," in *Proc. 25th Int. Workshop Power Timing Model. Optim. Simul.*, 2015, pp. 103–110.

[107] I. Ahmad, S. Ranka, and S. U. Khan, "Using game theory for scheduling tasks on multi-core processors for simultaneous optimization of performance and energy," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, 2008, pp. 1–6.

[108] M. Abdeyazdan, S. Parsa, and A. M. Rahmani, "Task graph pre-scheduling, using nash equilibrium in game theory," *J. Supercomput.*, vol. 64, no. 1, pp. 177–203, Jan. 2013.

[109] J. Wilkins, I. Ahmad, H. Fahad Sheikh, S. Faheem Khan, and S. Rajput, "Optimizing performance and energy in computational grids using non-cooperative game theory," in *Proc. Int. Conf. Green Comput.*, 2010, pp. 343–355.

[110] B. Yang, Z. Li, S. Chen, T. Wang, and K. Li, "Stackelberg game approach for energy-aware resource allocation in data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 12, pp. 3646–3658, Dec. 2016.

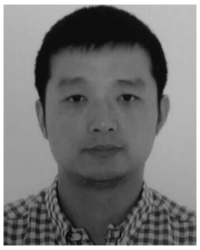
[111] B. Yang, Z. Li, and S. Jiang, "Cooperative game approach for energy-aware load balancing in clouds," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Appl. IEEE Int. Conf. Ubiquitous Comput. Commun.*, 2017, pp. 9–16.

[112] J. Yang, B. Jiang, Z. Lv, and K.-K. R. Choo, "A task scheduling algorithm considering game theory designed for energy management in cloud computing," *Future Gener. Comput. Syst.*, vol. 105, pp. 985–992, 2020.

[113] G. Xie *et al.*, "Minimizing redundancy to satisfy reliability requirement for a parallel application on heterogeneous service-oriented systems," *IEEE Trans. Services Comput.*, vol. 13, no. 5, pp. 871–886, Sep./Oct. 2020.

[114] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Comput. Surv.*, vol. 52, no. 6, Jun. 2019, Art. no. 125.

Guoqi Xie (Senior Member, IEEE) received the PhD degree in computer science and engineering from Hunan University, Changsha, China, in 2014. He is currently a professor at Hunan University, China. He was a postdoctoral research fellow with Nagoya University, Japan. His current research interests include real-time systems, automotive embedded systems, embedded safety, and security. He received the 2018 IEEE TCSC Early Career Researcher Award. He is currently serving on the editorial boards of the *Journal of Systems Architecture*, the *Microprocessors and Microsystems*, the *Journal of Circuits, Systems and Computers*. He is an ACM senior member.



Xiongren Xiao received the PhD degree in computer science and engineering from Hunan University, Changsha, China, in 2014. He is currently an assistant professor at the College of Computer Science and Electronic Engineering, Hunan University, China. His current research interests include low-energy scheduling, embedded and cyber-physical systems, and safety- and security-critical systems.

Hao Peng received the PhD degree in computer science and engineering from Hunan University, Changsha, China, in 2020. His current research interests include low-energy scheduling, embedded and cyber-physical systems, parallel, and distributed systems.

Renfa Li (Senior Member, IEEE) is currently the professor and chair in embedded and cyber-physical systems, Hunan University. He is also the chair of the Key Laboratory for Embedded and Cyber-Physical Systems. His major interests include computer architectures, embedded computing systems, cyber-physical systems, and Internet of things. He is a member of the council of the CCF, and a senior member of the ACM.

Keqin Li (Fellow, IEEE) is currently a SUNY distinguished professor of computer science at the State University of New York. He is also a distinguished professor with 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 more than 760 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He has served on the editorial boards of the *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Services Computing*, and *IEEE Transactions on Sustainable Computing*.