Energy-aware online task dispatching and scheduling for edge systems with energy harvesting

Mu Yuan and Nikolaos M. Freris

Abstract—**In this paper, we consider the problem of online task dispatching and scheduling in a system of devices that may possess energy harvesting capabilities. The objective is twofold, namely to maximize the cumulative weight of tasks that can be completed before their deadlines and to minimize the total energy consumption. Our proposed solution, termed ELISE, operates in an online fashion in that for each newly arriving task it decides between three alternatives (execute before another previously scheduled task, replace an existing task, or place in the waiting line) so as to meet the objectives. We analyze the complexity of ELISE and further provide performance guarantees in terms of bounds on the gap to optimality with regards to the two objectives. Extensive simulations attest to superior aggregate weight, energy consumption, guarantee ratio, and energy consumption per task, over baseline algorithms.**

Index Terms—**Energy-aware scheduling, energy harvesting, task allocation, replacement**

I. Introduction

The rapid proliferation of portable appliances, such as smart phones and sensors, has ushered the realm of device-oriented applications. On this front, notable is the advancement of real-time systems that feature stringent time execution constraint, (e.g., autonomous driving, unmanned aerial vehicles, smart grids, etc.) [1]. A popular paradigm lies in Hard Real-Time (HRT) applications, where it is critical to assure execution of tasks before some predetermined deadlines.

Task scheduling in HRT systems emanated from the pioneering work of Liu and Layland [2] on multi-task scheduling on a single processor, where the key concepts of rate monotonic and earliest-deadline-first scheduling were introduced. Since then, there has been a large body of methods for scheduling tasks on multiple devices. Archetypally, scheduling problems are combinatorial, whence it is notoriously challenging to obtain exact optimal solutions and thus the aim is to acquire suboptimal schedules of 'high quality'. The quality is typically assessed by a *competitive ratio*, defined as the ratio between the value of a certain performance metric achieved by the considered algorithm over the theoretically optimal value. For example, the problem of minimizing flow time (the difference between the completion and release time of a job) was studied in [3]: both offline and online algorithms were proposed, that attain improved bounds on the competitive ratio.

Scheduling of tasks in a distributed system of heterogeneous devices has been studied in [4] with different objectives: low time complexity, minimum task execution time, and low energy consumption. Another setting assumes precedence constraints [5] within the set of tasks, that is, there exists a strict execution order between different tasks. In the context of cloud computing, where tasks have to be scheduled on different devices (e.g., smartphones), energy consumption [6] becomes a key consideration for a number of reasons such as reducing operating costs, increasing system reliability, and promoting environmental friendliness. Last but not least, in the fast-emerging setting of edge computing, a set of edge servers are deployed near the mobile devices in a fashion that allows them to offload jobs with low latency. In such scenario, it becomes imperative to dispatch and schedule jobs so as to minimize the response time (defined as the elapsed time between the release of a job and the arrival of the computation result at the mobile device) [7].

The focus of the aforementioned works is on reducing energy consumption and minimizing execution time, under the assumption that the energy for all devices is sufficient. Nonetheless, in emerging applications involving devices that dynamically harvest energy from the environment, such an assumption is not appropriate. We illustrate via a real-life scenario, where a large number of low-power sensors are deployed in a subway station in Xi'an, China. The deployment is carried in an infrastructure-less manner (i.e., without power supply). In such context, edge devices need to harvest enough energy from various sources (e.g., light, wind, vibration, etc.) so as to conduct measurement of environmental parameters (such as temperature, humidity, pressure, gas concentration, etc.). The purpose of the system is to ascertain accurate real-time monitoring and detection of abnormal conditions. In this scenario, sensing tasks have variable weight (e.g., a safety-critical measurement has larger weight) and need to be dispatched and scheduled to devices in consideration of their energy budgets. This motivates the development of ELISE, which is a solution tailored for energy-critical edge systems with dynamic energy harvesting from the environment and a threefold objective: high response time, high guarantee ratio, and low energy consumption.

The setting of this paper assumes tasks with different importance rankings that are released by users in real-time, and feature heterogeneous execution times and energy consumption when run on different devices. Furthermore, unlike previous

The authors are with the School of Computer Science, University of Science and Technology of China, Hefei, 230000, China. Emails: yuanmu@mail.ustc.edu.cn,nfr@ustc.edu.cn.

This work was supported by the Ministry of Science and Technology of China under grant 2019YFB2102200. Correspondence to N. Freris.

methods, we specifically consider the case where devices may harvest energy from the environment. We address the high variability of the energy harvesting process by maintaining a dynamic energy budget for each device. We proceed to develop solutions that aim to minimize the energy consumption and maximize the weight of tasks completed before the deadline. The joint scheduling and dispatching problem is formulated as a multi-objective optimization problem. Obtaining an exact solution is hard, therefore, we propose an efficient greedy method called ELISE (Energy-aware onLine dIspatching & SchEduling) to obtain and update real-time schedules. The main considerations in designing our proposed method are summarized as follows:

• **time constraints:** In an HRT system, each task has a deadline, and tasks that are not completed before their respective deadlines are dropped.

• **energy constraints:** Each device requires to consume a certain amount of energy to perform a given task. Therefore, for each task, in addition to meeting the deadline it is also necessary to ensure that the device it is assigned to has sufficient energy for its execution.

• **energy harvesting:** Devices may extract energy from the environment in real-time. The harvested energy is uncertain and time-varying and its variability has to be factored in during the operation of the whole system.

To the best of our knowledge, this is the first work to study the online joint dispatching and scheduling problem in a dynamic energy harvesting system, while also explicitly considering the importance rankings of different tasks. The main contributions of this paper enlist:

1) We formulate a novel light-weight energy-aware dispatching and scheduling problem in an energy harvesting system, where different tasks have different weights. Our formulation considers dynamic energy budgets and a twofold objective: to maximize the total weight of completed tasks and to minimize the total energy consumption.
2) We propose ELISE, a light-weight energy-aware dispatching and scheduling algorithm. For each incoming task, the proposed method determines its schedule in a greedy manner: it decides whether to replace an existing task or place it ahead of another already scheduled task, in order to meet the objectives. The dispatching algorithm selects the device to execute the newly arriving task so that the aggregate weight, energy consumption, guarantee ratio (GR), and energy consumption per task (ECT) are optimized.
3) Our proposed solution is suitable for real-time operation: a worst-case linear run-time in the number of tasks and devices is established for each instance of dispatching/scheduling decisions.
4) We analyze the competitive ratio (i.e., the ratio between the attained and optimal values) for total weight and energy consumption, and establish bounds that depend solely on the system heterogeneity (in terms of weight, execution time, and energy requirement).
5) To demonstrate the performance of our proposed algorithm, we compare it with energy-aware extensions of popular baseline algorithms, namely e-FCFS, e-EDF, and e-HWF. Our experimental results show that ELISE outperforms all methods in terms of both reduced energy consumption as well as higher total weight pertaining to the completed tasks.

The rest of the paper is organized as follows. Related art is discussed in Section II. Section III introduces the system model and Section IV the problem formulation. Our proposed method is described and analyzed in Section V. The experimental findings are presented in Section VI, while Section VII concludes the paper.

II. Related work

Scheduling a set of tasks in a distributed system of heterogeneous devices was studied in [4], where the authors proposed a method termed EMuWS to determine a set of inefficient processors for shut-down in order to reduce computing resources. In addition, [8] modeled the resource allocation problem as a bi-objective optimization problem of reducing energy consumption and makespan (the completion time of the last task). To achieve energy-efficient computation offloading under hard constraints on completion time, [9] provided a policy called eDors.

Another setting considers precedence constraints within the set of tasks, for which the methods HEFT [5] and CPOP [5] were shown to significantly improve the makespan. A HEFT-based approach was adopted in [10] for reducing energy consumption while achieving a low makespan, while a two-phase solution named PASTA [11] was proposed to reduce the overall makespan and aggregate energy consumption. Additionally, there have been approaches based on approximation algorithms [12]. Albeit their efficacy in reducing the energy consumption of schedules, the main limitation lies in substantially higher execution times compared to simpler alternatives, which renders them inappropriate for Hard Real-Time scheduling in large systems. [13] presents the design, implementation, and evaluation of EScheduler, an energy-efficient soft real-time CPU scheduler for multimedia applications running on a mobile device. There has also been work developing randomized scheduling policies such as in [14], and [15]; nevertheless, none of these methods incorporates hard deadlines.

For scheduling a set of tasks in cloud computing and edge computing, a heuristic approach called EAS [6] was proposed to address the issue of energy consumption optimization. Besides, [16] devised a rolling-horizon architecture for real-time task scheduling in virtualized clouds alongside an algorithm named EELISEH for handling real-time, aperiodic, and independent tasks. Dynamic flow scheduling in data center networks was investigated in [17], where an algorithm named DEDFS was proposed. An online method called OnDisc [7] was introduced to minimize the total weighted response time over all jobs, while another method called Dedas [18] was proposed to meet the maximum number of deadlines while considering management of the networking bandwidth and computing resources. [19] considered migration (transfer of an unfinished task to another machine) in heterogeneous machine settings, and proposed a method to maximize the total obtained profit (as specified by an introduced metric on the completion time of the jobs).

In brief, the main focus of prior art is on reducing energy consumption and minimizing execution time, under the assumption that the energy for all devices is sufficient. In contrast, in this paper, we develop methods for edge systems with *energy harvesting* capabilities. This scenario involves limited and dynamic energy budgets for executing tasks, and our proposed solution (ELISE) has a dual objective of maximizing the total weight of tasks that can be finished before their deadlines and minimizing energy consumption, while featuring low run-time suitable for online implementation.

III. System model

We consider a system that comprises of a set $\mathcal{D}$ of heterogeneous devices which may extract energy from the surrounding environment. Users release tasks (we use the terms 'task' and 'job' interchangeably) in real-time, and we denote with $\mathcal{J}$ the set of tasks released from the users. We adopt the most general scenario in which: a) jobs can appear in arbitrary order and times, b) the energy harvested by the devices is uncertain and variable with time. In view of high migration overheads in real systems [20], a device is not permitted to migrate jobs to others after the job dispatching decision is taken.

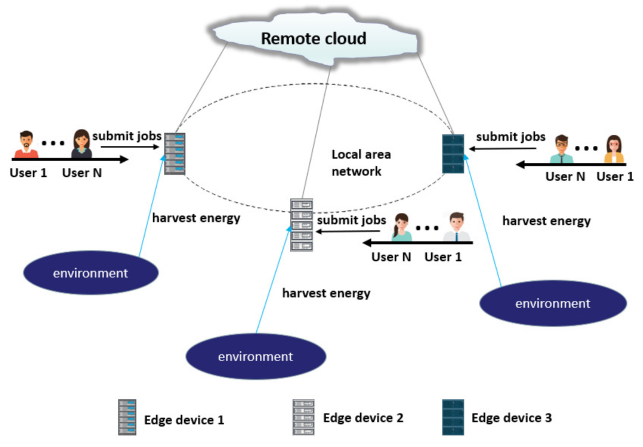

Fig. 1. Overview of the system model.

Parameter Description: We denote the set of heterogeneous devices as $\mathcal{D}$ and the set of tasks released from the users as $\mathcal{J}$. We use n for the total number of heterogeneous devices, R for the number of tasks, t for the current time slot, and T for the total number of time units over which system operation is considered. A new task arriving in the system is represented with the following set of parameters: we denote by r_i the release time of task i, d_i is its *hard* deadline, $up_{i,j}$ is the time required for task i to upload to device j, $do_{i,j}$ the time required for device j to return the result of task i to the user, and w_i represents the weight of task i (which captures its importance). Furthermore, we define variables related to the dispatching/scheduling decisions as follows. We denote with st_i the start execution time of task i and ft_i the finish time, i.e., the time when the execution result of the task is returned to a user (the hard deadline requirement is $ft_i \leq d_i$). We use ls_i as the latest start time of task i (i.e., the latest time that the task can start execution and still meet the deadline; this is given by the deadline minus the sum of upload, execution, and download times). We denote $wt_{i,j}$ as the waiting time for task i to run on device j. In addition, we denote $et_{i,j}$ as the execution time of task i on device j (a task is executed on a single device, and cannot be migrated to other devices, so a unique device j is chosen by the allocation algorithm for every given task i). The following relations are satisfied by definition: $st_i = r_i + up_{i,y_i} + wt_{i,y_i}$ and $ft_i = st_i + et_{i,y_i} + do_{i,y_i}$, where y_i corresponds to the device that task i is assigned to. The total delay of task i running on device j is represented by $de_{i,j}$, which includes upload time, download time, and waiting time. We denote $ec_{i,j}$ as the energy consumption of task i on device j per unit time (this is simply set equal to the energy consumption rate of the device, ecr_j). For each device j, we use $e_j(t)$ to denote its remaining energy at time t. Last, we use $eh_j(t)$ to denote the energy harvested by device j at time slot t. All parameters are summarized in Table I, while the system overview is illustrated in Figure 1.

TABLE I
Notation

Parameters	Meaning
$\mathcal{D}$	set of heterogeneous devices
$\mathcal{J}$	set of jobs released by users
n	number of heterogeneous devices
R	number of released tasks
t	current time
T	total time of system operation
r_i	release time of task i
d_i	hard deadline of task i
$up_{i,j}$	time required for task i to upload to device j
$do_{i,j}$	time required for device j to return the result of task i
w_i	weight of task i
st_i	start execution time of task i
ft_i	time when the task i is returned to the user
ls_i	latest start time of task i
$wt_{i,j}$	waiting time for task i to run on device j
$et_{i,j}$	execution time of task i on device j
y_i	index of device which runs task i
$de_{i,j}$	the total delay of task i running on device j
$ec_{i,j}$	energy consumption of task i on device j per unit time
ecr_j	energy consumption rate of device j per unit time
$e_j(t)$	remaining energy of device j at time t
$eh_j(t)$	energy harvested by device j at time slot t

IV. Problem Formulation

In view of the aforementioned model, our goal is to design an algorithm to schedule newly-incoming tasks to devices that harvest energy from the environment aiming to satisfy two key objectives: a) maximize the total weight of tasks that can be finished before their deadlines during the given period, and b) minimize the total energy consumption under the condition of a). This can be modeled as a bilevel constrained optimization problem (where the purpose is to select a schedule that satisfies the second objective from the set of schedules that satisfies the first) as follows:

$$\textbf{Obj. 1:} \underset{x_{i,j}(t)}{\text{maximize}} \sum_{i \in \mathcal{J}} w_i \cdot \mathbf{1}\{ft_i \leq d_i\}$$

$$\textbf{Obj. 2:} \underset{x_{i,j}(t)}{\text{minimize}} \sum_{t \in [T], i \in \mathcal{J}, j \in \mathcal{D}} ec_{i,j} \cdot x_{i,j}(t)$$

Constraints:

$$x_{i,j}(t) \in \{0,1\}, \quad \forall t \in [T], i \in \mathcal{J}, j \in \mathcal{D} \tag{1}$$

$$\sum_{i \in \mathcal{J}} x_{i,j}(t) \leq 1, \quad \forall t \in [T], \forall j \in \mathcal{D} \tag{2}$$

$$\sum_{j \in \mathcal{D}} x_{i,j}(t) \leq 1, \quad \forall t \in [T], \forall i \in \mathcal{J} \tag{3}$$

$$ft_i = \max\{t | x_{i,y_i}(t) = 1\}, \quad \forall i \in \mathcal{J} \tag{4}$$

$$e_j(t) = e_j(t-1) + eh_j(t) - \sum_{i \in \mathcal{J}} ec_{i,j} \cdot x_{i,j}(t), \forall j \in \mathcal{D} \tag{5}$$

The first objective corresponds to maximization of the total weight over all completed tasks, while the second to the minimization of the total energy consumption. We use $[T]$ to abbreviate $\{1, \ldots, T\}$; $x_{i,j}(t)$ is a binary variable that equals 1 if the device j is processing task i at time t, and 0 otherwise; y_i represents the device to which task i is assigned by the allocation algorithm (if task i is rejected by the algorithm, we simply set $y_i = 0$ by convention, whence also $ft_i = \infty$); $\mathbf{1}\{\cdot\}$ denotes the indicator function, that is, $\mathbf{1}\{ft_i \leq d_i\}$ equals 1 if $ft_i \leq d_i$ and 0 otherwise.

Constraint (1) simply prescribes that $x_{i,j}(t)$ is a binary variable, as introduced in the prequel; constraint (2) guarantees that each device processes at most one task at any given time; constraint (3) indicates that each task can only be processed on at most one device at any given time; constraint (4) defines the value of ft_i: for completed tasks it equals the completion time, otherwise the value infinity is assigned; constraint (5) defines the evolution of energy budget of a device $e_j(t)$: the energy budget is increased by the energy harvested and decreased by the cost of executing a task on the device.

In addition to the intractability of the bi-objective optimization problem, an additional challenge lies in the unpredictability of the harvested energy. That is, the dispatcher/scheduler does not have access to future information about harvested energy values and needs to carry decisions online, based solely on the available energy budgets at the time of decision. To simplify the problem, we suppose that the execution time and the energy consumption for each task on different devices are known in advance. For this purpose, we develop a greedy online mechanism as elaborated next.

V. Algorithm

In this section, we provide the description of ELISE, alongside the analysis of its complexity and competitive ratio.

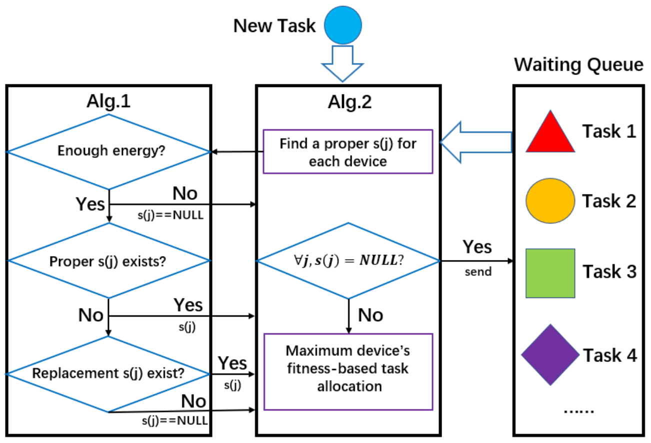

Fig. 2. Workflow of ELISE.

A. Brief introduction

The operation of ELISE can be divided into two parts. The purpose of the first part is to find the best scheduling scheme possible when dispatching a certain task to a certain device. The mechanism comprises of three stages: 1) use the sum of currently available energy as the initial energy conditions for scheduling tasks (judge whether the device is expected to have sufficient energy); 2) if the energy is sufficient, find a proper scheme that can ensure all tasks in this device can be finished before their deadlines; 3) if there is no such scheme, compute a proper replacement scheme.

The purpose of the second part is to decide, for each task, the device to allocate it to. The proposed scheme adopts a fitness function (defined as the negative of the total energy consumption of the scheme), whence among scheduling schemes that satisfy Obj. 1, the scheme with the highest fitness will be selected as the final scheme. The basic workflow of our method is illustrated in Figure 2. We proceed to explicate and motivate our design choices in what follows.

B. Design choices

ELISE consists of two components: allocation & replacement (Algorithm 1), and joint dispatching & scheduling (Algorithm 2).

Algorithm 1 (Allocation & Replacement) is used to provide an effective mechanism for assigning new tasks to a specified device. We denote the current scheduling plan as s (line 1). The algorithm proceeds to check whether the energy is sufficient for the completion of the new task (lines 2-4).

If so, it computes a schedule bearing two objectives: a) the new task has the lowest possible priority, and b) all tasks following the new task (including this new task) can be completed before their respective deadlines (lines 5-16). When it is infeasible to find a scheme to guarantee that all tasks meet their deadlines, replacement is explored: our method computes the latest start time of the new task (line 18), finds a replacement candidate set (line 19), and selects the task with the minimum weight that has higher energy cost than the new task for possible replacement (line 20). If the selected task has lower weight than the new one, then it is replaced and finish times of all tasks behind it are updated, otherwise no replacement is carried (lines 21-30).

Algorithm 1 Allocation & Replacement

```
Input: task $i$, device $j$, current scheduling plan $s$
Output: new scheduling plan $s^*$
1: $s = \{N_1, \ldots, N_k\}$; $\{1, \ldots, k$ represent the indices of tasks that have not yet started executing$\}$
2: if $\sum_s (ec_{s,j} \cdot et_{s,j}) + (ec_{i,j} \cdot et_{i,j}) > e_j(t)$ then $\{$if energy is not enough, algorithm ends $\}$
3:     return NULL;
4: end if
5: flag=false; $\{$'flag' is used to judge whether there exists a suitable insertion position$\}$
6: for $\gamma$ = $k+1$ to 1 do $\{\gamma$ represents the insertion position of the new task $i\}$
7:     produce schedule $s^*$ according to $\gamma$ & compute $ft_i$;
8:     $J^* = set\,of\,all\,tasks\,after\,task\,i$;
9:     for each task $l$ in $J^*$ do $\{$update finish time for all tasks$\}$
10:        $ft_l^* = ft_l + up_{i,j} + et_{i,j} + do_{i,j}$;
11:    end for
12:    if $\forall l \in J^*, ft_l^* \le d_l$ and $ft_i \le d_i$ then $\{$judge whether the new schedule serves all tasks before their deadlines$\}$
13:        use $s^*$ as the final schedule and update the finish times of all tasks (after and including task $i$);
14:        flag=true; break;
15:    end if
16: end for
17: if flag==false then
18:    $ls_i = d_i - up_{i,j} - et_{i,j} - do_{i,j}$;
19:    $J^\kappa = \{N_l | st_{N_l} < ls_i, l = 0, 1, \ldots, k\}$;$\{$find the set of tasks which has lower start time than the latest start time of the new task$\}$
20:    $q = \arg\min_{w_l} \{l \in J^\kappa | (ec_{l,j} \cdot et_{l,j}) > (ec_{i,j} \cdot et_{i,j})\}$
21:    if $w_i > w_q$ then
22:        replace task $q$ with task $i$; send task $q$ into the waiting line;
23:        for each task $l$ in $J$ behind $q$ do $\{$update finish time of the corresponding tasks$\}$
24:            $ft_l += up_{i,j} + et_{i,j} + do_{i,j} - (up_{q,j} + et_{q,j} + do_{q,j})$;
25:        end for
26:        produce $s^*$ from $s$;
27:    else
28:        $s^* = NULL$;
29:    end if
30: end if
31: return $s^*$;
```

Algorithm 2 Joint Dispatching & Scheduling

```
Input: task $i$, current time $t$
1: $\forall l \in \mathcal{D}$, use Algorithm 1 to produce schedule $s_l^*$;
2: Construct $s_{final}^*$ based on $\operatorname{argmax}_{l \in \mathcal{D}} fit(s_l^*)$;$\{fit(s_l^*)$ is defined as the negative of the total energy consumption of schedule $s_l^*\}$
3: if $s_{final}^* = NULL$ then $\{$if the task cannot be scheduled to any device$\}$
4:     send task $i$ into the waiting line;
5:     for every given time interval $I$ do
6:         if $t + \min_{j \in \mathcal{D}} \{up_{i,j} + et_{i,j} + do_{i,j}\} > d_i$ then
7:             reject task $i$;
8:         else
9:             go to step 1;
10:        end if
11:    end for
12: else
13:    $\forall k \in \mathcal{J}$, update parameters according to $s_{\text{final}}^*$;
14:    Follow the chosen schedule $s_{\text{final}}^*$ to schedule tasks;
15: end if
```

Algorithm 2 (Joint Dispatching & Scheduling) is used to dispatch tasks to devices. When a new task is released, for each device, Algorithm 2 is invoked to generate a scheduling scheme. The fitness values of these scheduling schemes are computed and the device with the highest value is chosen for dispatching, which is a locally optimal strategy (line 1-2). In case that all the scheduling schemes are infeasible, then the new task is placed into the waiting line (line 4), because it is not advisable to simply discard them outright (The reason is that it may still be possible to execute them before their deadlines, e.g., if the device collects enough energy from the environment, and there are no other tasks to be performed). For tasks in the waiting line, after each I time units, we use step 1 to reschedule them or reject them if they can not meet their deadline (lines 5-11). If a proper schedule can be found, its parameters are updated (lines 13-14).

C. Complexity

The time complexity of ELISE is in what follows.

Lemma 1 *The time complexity of dispatching a task is at most $\mathcal{O}(r + n \log \frac{r}{n})$, where r denotes the total number of tasks scheduled but not yet running, and n denotes the total number of devices.*

Proof: Suppose $r = r_1 + r_2 + \ldots + r_n$ (where n denotes the number of devices, and r_i denotes the number of tasks scheduled but not yet running on device i, $i = 1, 2, \ldots, n$). We consider the worst case that, for each device, the loop (line 6 of Algorithm 1) is fully executed. The time complexity of line 10 is r_i $(i = 1, 2, \ldots, n)$, the time complexity of line 12 is $2r_i$, whence the total time complexity for device i is $\mathcal{O}(r_i)$, therefore the total time complexity of this loop is $\mathcal{O}\left(\sum_{i=1}^n r_i\right) = \mathcal{O}(r)$. If a schedule is not returned after the end of this loop, replacement is considered (lines 17-

30 of Algorithm 1). In this phase, we first construct the candidate set in line 19 (using dichotomy), which can be done in $\mathcal{O}(r_i+\log r_i)$ for each device i. After finding the candidate set, the time complexity of lines 20 and 23 in Algorithm 1 is at most $\mathcal{O}(r_i)$, so that the total time complexity of this phase is $\mathcal{O}\left(\sum_{i=1}^{n}(r_i+\log r_i)\right)$, which simplifies to $\mathcal{O}\left(r+nlog\frac{r}{n}\right)$ in view of the geometric mean/arithmetic mean inequality $r_1\cdot r_2\cdots r_n\leq(\frac{r_1+r_2+\ldots+r_n}{n})^n$. Adding the complexities of the two phases completes the proof. ■

Corollary 1 *Suppose there are R tasks during the whole time period. If every new task can be allocated to a device (i.e., no task will be sent into the waiting line), the total time complexity of the proposed method is at most $\mathcal{O}(R^2+Rn\log\frac{R}{n})$.*

Proof: From Lemma 1, it follows that the time complexity of dispatching a task is at most $\mathcal{O}(r+n\log\frac{r}{n})$. The worst-case total time complexity for R tasks is given by

$$\mathcal{O}\left(\sum_{r=1}^{R}(r+n\log r-n\log n)\right)$$
$$=\mathcal{O}\left(\frac{R(R+1)}{2}-Rn\log n+n\log R!\right)$$

whence the result follows using: $R!\leq R^R$. ■

D. Theoretical analysis

Our analysis serves to provide bounds on the competitive ratio (i.e., the ratio of the value attained by ELISE and the optimal value) for the two objectives in consideration, namely: aggregate weight and energy consumption. Our analysis considers the worst case.

Theorem 1 *Denote by $w_{\min}$, $w_{\max}$ respectively the minimum and maximum weight, and by $et_{\min}$, $et_{\max}$ the minimum/maximum execution time across all tasks and devices. The competitive ratio on total weight for ELISE is lower bounded as:*

$$\frac{w}{w^\star}\geq\frac{et_{\min}\cdot w_{\min}}{et_{\max}\cdot w_{\max}}.$$

Proof: ELISE only substitutes a task with lower energy consumption and higher weight for a task with higher energy consumption and smaller weight. Let p denote the percentage of tasks that can be completed before their deadlines by ELISE and $p^\star$ the optimal value. In the worst case, there is no chance to do replacement, whence, it holds that $w\geq Rpw_{\min}$ and $w^\star\leq Rp^\star w_{\max}$, therefore

$$\frac{w}{w^\star}\geq\frac{p\cdot w_{\min}}{p^\star\cdot w_{\max}}.$$

We proceed to bound $\frac{p}{p^\star}$. In the worst case, the new task has execution time $et_{\max}$, and replaces a task with execution time $et_{\min}$. Then in the remaining time duration pertaining to all replaced tasks, the number of tasks that can be completed is at most $\frac{Rp(et_{\max}-et_{\min})}{et_{\min}}$, whence

$$\frac{p}{p^\star}=\frac{Rp}{Rp^\star}\geq\frac{Rp^\star-\frac{Rp(et_{\max}-et_{\min})}{et_{\min}}}{Rp^\star}=1-\left(\frac{et_{\max}}{et_{\min}}-1\right)\frac{p}{p^\star},$$

which simplifies to

$$\frac{p}{p^\star}\geq\frac{et_{\min}}{et_{\max}}.$$

Combining this with $\frac{w}{w^\star}\geq\frac{p\cdot w_{\min}}{p^\star\cdot w_{\max}}$ completes the proof. ■

Theorem 2 *Denote with e_{min}, e_{max} the minimum and maximum total energy consumption, and with et_{min}, et_{max} the minimum and maximum execution time (over tasks and devices). In the worst case, the ratio between the total energy (e) consumed by ELISE and the optimal value ($e^\star$) satisfies:*

$$\frac{e}{e^\star}\leq\frac{et_{max}\cdot e_{max}}{et_{min}\cdot e_{min}}.$$

Proof: Following Obj. 2, both ELISE and the optimal solution share the property of no partial execution (i.e., a task will be executed only if it is guaranteed to complete). With an analogous argument as in the proof of Theorem 3, it holds that $e\leq Rpe_{max}$ and $e^\star\geq Rp^\star e_{min}$, therefore, $\frac{e}{e^\star}\leq\frac{p\cdot e_{max}}{p^\star\cdot e_{min}}$ In a similar fashion as in the proof of Theorem 1 (by swapping the role of ELISE and the optimal solution) we obtain $\frac{p}{p^\star}\leq\frac{et_{max}}{et_{min}}$, , whence combining concludes the proof. ■

VI. Experimental validation

To demonstrate the merits of ELISE, we quantitatively compare it with three of the most widely used scheduling algorithms, namely: FCFS (First come first serve), EDF (Earliest deadline first), and HWF (Highest weight first). We note that these algorithms operate based on the assumption of sufficient energy, i.e., they are not tailored for situations of energy shortage. To be fair in our comparisons, we modify these widely used traditional algorithms to adapt to the situation of energy shortage. The modified energy-aware variants that constitute the baselines in our experiments are described in detail in the following.

e-FCFS: when a new task comes, for each device, the modified energy-aware variant of FCFS checks whether the energy conditions and deadline conditions are met or not. If not, the task is sent to the waiting line. When there are more than one devices that meet the condition, it is assigned to the one with the lowest energy consumption for the given task. Incoming tasks and tasks in the waiting line are then served according to FCFS principle.

e-EDF: Newly arriving tasks and tasks in the waiting line are sorted by their deadlines, and a task with the earliest deadline is scheduled first; the same modification as in e-FCFS is adopted.

e-HWF: Newly arriving tasks and tasks in the waiting line are sorted by their weights, and a task with the largest weight is scheduled first; the same modification as in e-FCFS is adopted.

The performance metrics used to evaluate the system performance include:

1. Guarantee ratio (GR) is defined as the ratio of the total count of tasks guaranteed to meet their deadlines over an upper bound on the number of tasks that can be timely completed.
2. Total energy consumption is the total energy consumed for executing the tasks.
3. Total weight is the aggregate weight of the tasks executed.

4. Energy consumption per task is calculated as ECT=Total energy consumption/Completed task count.

A. Simulation Method and Parameters

To ensure reproducibility of the experiments, we make the following design choices in evaluating the performance of the aforementioned algorithms. The parameter, and the values (scaled units) used in the experiments are detailed in Table II.

TABLE II
SIMULATION PARAMETERS

Parameter	Meaning	Value(Fixed)-(Varied)
T	time range of system operation	100,000
$R(10^4)$	total number of tasks	5(6,7,8,9)
n	total number of devices	10
$d_{\max}$	largest deadline coefficient	50
d_b	base relative deadline	40(20,60,80,100)
$w_{\max}$	largest weight	15
$E_{\max}$	largest energy harvesting rate	8(2,4,6,10)
$\ell_{\max}$	largest task size	40(20,60,80,100)
$\ell_{\min}$	smallest task size	2
I	time interval for re-sche tasks	10
ecr	energy consumption rate	U(1,3)
s	speed of the device	U(1,2)
sr	stay_rate	0.5
E_s	upper bound of stored energy	500

1. Each device is modeled to have a single-core cpu with a speed varying from 1 to 2 (scaled units).
2. The energy consumption rate of the different kinds of devices can vary from 1 to 3 (scaled units).
3. The energy harvested by each device at every time unit varies with time and is assumed i.i.d. uniformly distributed $U(0, E_{\max})$, where $U(\cdot)$ represents the discrete uniform distribution.
4. The execution time (for simplicity, we have set upload/download times to 0) and energy consumption of task i on device j are defined as follows:

$$et_{i,j} = \frac{\text{size of task } i}{\text{speed of device } j},$$
$$ec_{i,j} = ecr_j.$$

5. The deadline and the weight of a task i are drawn as follows:

$$d_i = r_i + \max_j et_{i,j} + d_b + U(0, d_{\max}),$$
$$w_i = U(1, w_{\max}).$$

6. The parameter sr is used to determine the probability of tasks arriving at the same time.

To simplify the simulation of this algorithm, our simulation environment is structured as follows: any task released by the user is first sent to the scheduler, which is responsible for executing ELISE to decide which device the task is assigned to, or send it to the waiting queue or discard it. The operation is centralized and the delay for the devices to communicate with each other is short and based on this assumption, we ignore the time required for the communication between devices (e.g., task upload and download times, etc.), and the scheduler will discard tasks that cannot be completed before the deadline.

B. Performance Impact of Task Count

In this section, we present a series of experimental results to assess the performance comparison between our proposed method and the three selected baselines in terms of the impact of task count. Fig. 3 illustrates the results.

As it can be observed from Fig. 3(a), all methods evaluated feature a stable growth in total weight with the increase of task count. This is because when the energy is sufficient and the deadlines are not too tight, it holds that the higher the number of tasks the more tasks can be completed, thus the algorithm attains a higher aggregate weight. Nevertheless, it is found that ELISE can achieve substantially higher total weight compared to the baselines; this is attributed to the replacement strategy of ELISE that is efficient in swapping tasks with higher energy consumption and lower weight with tasks with lower energy requirement and higher weight.

From Fig. 3(b), it can be inferred that, as expected, the total energy consumption increases with the task count. Fig. 3(b) shows that all algorithms consume almost the same amount of energy, with ELISE featuring slightly higher energy consumption. This is a positive attribute, in view of the much higher Guarantee Ratio (see Fig. 3(c)) achieved by ELISE which completes more tasks with almost the same amount of energy. From Fig. 3(d), we deduce a much lower ECT, i.e., a superior energy utilization. To conclude, the positive findings vis-à-vis higher task completion ratio and energy economy are due to employing insertion and replacement policy which proves to be an effective means when scheduling real-time tasks.

C. Performance Impact of Base Deadline

To examine the impact of the base deadline on performance, we vary the base deadline from 20 to 100. Fig. 4 illustrates the performance of the four algorithms in terms of total weight, energy consumption, guarantee ratio, and consumed energy per completed task.

From Figs. 4(a),(c),(d), we observe that ELISE substantially outperforms the baseline methods in terms of total weight, GR, and ECT. Fig. 4(c) reveals that for the baseline methods the Guarantee Ratio increases in the base deadline for values 20 to 40, but then decreases. This is because, for smaller values of the base deadline, an increase of d_b implies that the deadline becomes less stringent so that more tasks can be completed. However, when the base deadline is too large, it becomes more common for some tasks with long execution time to meet the deadline condition, and are therefore not sent to the waiting line but are executed first. As a result, the tasks with shorter execution time will not be executed first and the Guarantee Ratio eventually declines. On the contrary, our method features a consistent increase of Guarantee Ratio due to the involved insertion & replacement operations.

D. Performance Impact of task size

To examine the impact of the task size, we vary the value of $\ell_{\max}$ from 20 to 100. Fig. 5 illustrates the assessment of the performance metrics for all tested algorithms.

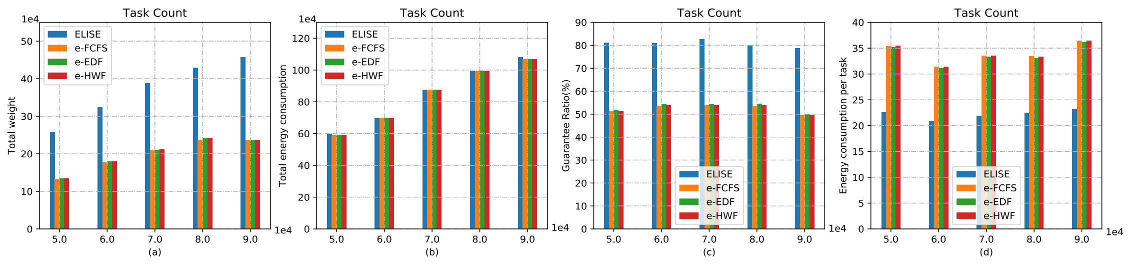

Fig. 3. **Performance Impact of Task Count.** We only vary the value of R from 5e4 to 9e4(5e4,6e4,7e4,8e4,9e4), the values of other parameters remain fixed (see Table II). The four subfigures from left to right represent the results of Total weight, Total energy consumption, Guarantee Ratio, and Energy consumption per task, respectively.

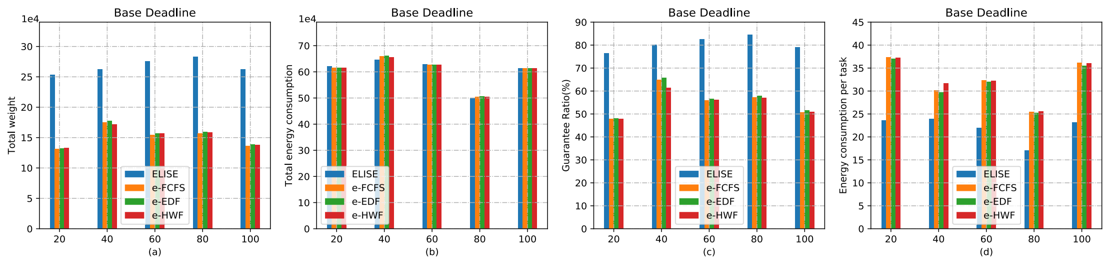

Fig. 4. **Performance Impact of Base Deadline.** We only vary the value of d_b from 20 to 100(20,40,60,80,100), the values of other parameters remain fixed (see Table II). The four subfigures from left to right represent the results of Total weight, Total energy consumption, Guarantee Ratio, and Energy consumption per task, respectively.

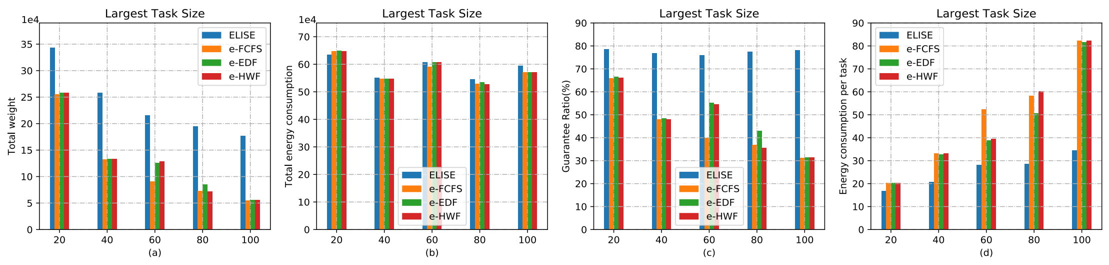

Fig. 5. **Performance Impact of Largest Task Size.** We only vary the value of $\ell_{\max}$ from 20 to 100(20,40,60,80,100), the values of other parameters remain fixed (see Table II). The four subfigures from left to right represent the results of Total weight, Total energy consumption, Guarantee Ratio, and Energy consumption per task, respectively.

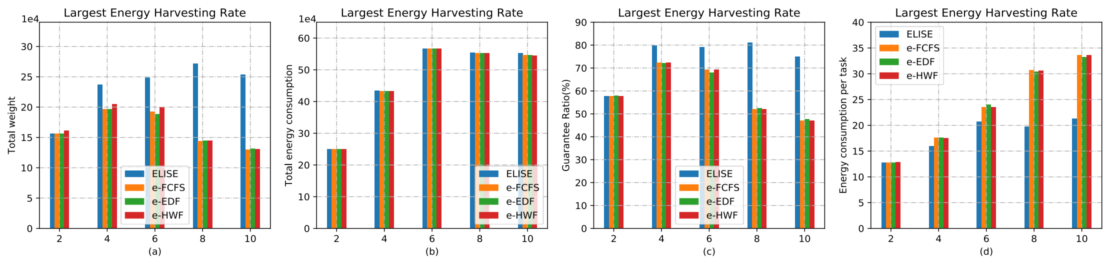

Fig. 6. **Performance Impact of Largest Energy Harvesting Rate.** We only vary the value of $E_{\max}$ from 2 to 10(2,4,6,8,10), the values of other parameters remain fixed (see Table II). The four subfigures from left to right represent the results of Total weight, Total energy consumption, Guarantee Ratio, and Energy consumption per task, respectively.

When $\ell_{\max}$ increases, it holds that the total count of tasks meeting their deadlines decreases, and the same holds for the upper bound of the total count of tasks that can be completed. Fig. 5(c) demonstrates that an increase in the task size leads to a consistent deterioration of the guarantee ratio for all baselines. Nonetheless, our proposed method does not suffer from such a phenomenon but rather features a notable insensitivity. This formidable aspect is once again attributed to its efficient replacement and insertion operations. Besides, as the task size becomes larger and larger, the energy consumption for each task increases.

E. Performance Impact of Energy Harvesting Rate

We vary the value of the energy harvesting rate from 2 to 10. Fig. 6 illustrates the performance of the four algorithms.

We observe that when the energy harvesting rate is very low, there is no apparent difference between the performance of the four algorithms. This is due to the fact that for very low rate of harvesting energy, most of the tasks remain in the waiting line: once the energy is sufficient, there will be only one task to be executed, whence the insertion and replacement operations will almost never be invoked. It is worth noting that the improvement gap of ELISE over the baselines is increasing in the energy harvesting rate, see Fig. 6(a). Fig. 6(c) indicates that for the three baseline algorithms, the guarantee ratio increases with the increase of energy harvesting rate initially, but then starts decreasing beyond a certain rate value. The reasons for this phenomenon are similar to the previous case: when the energy harvesting rate is too high, a large portion of tasks with high energy consumption meet the energy condition, whence they will not be sent to the waiting line, but will be executed first (because there are no insertion and replacement operations). As a result, the tasks with lower energy consumption will not be given priority. Due to a positive correlation between execution time and energy consumption, this implies that the tasks with shorter execution time are not executed first, which brings about the fact that the guarantee ratio eventually declines. Our method is impermeable to this phenomenon and provides higher stability to variations in energy harvesting rate.

VII. Conclusion

ELISE is a new method for task dispatching and scheduling in distributed systems with dynamic energy harvesting capabilities. Its low complexity makes it suitable for online execution in settings where tasks of variable importance are released in real-time, and feature heterogeneous execution time and energy consumption across devices. We have established bounds on the gap to optimality, in terms of both aggregate weight and energy consumption, that depend exclusively on the level of heterogeneity. The performance of ELISE was also analyzed via extensive simulations that corroborate an increase of the total weight of completed tasks as well as energy savings.

References

[1] C. S. Stangaciu, M. V. Micea, and V. I. Cretu, “Energy efficiency in real-time systems: A brief overview,“ in IEEE International Symposium on Applied Computational Intelligence & Informatics (SACI), pp. 275–280, 2013.

[2] C. L. Liu and J. W. Layland, “Scheduling algorithms for multiprogramming in a hard-real-time environment,” Journal of the ACM, vol. 20, pp. 46–61, 1973.

[3] N. Garg and A. Kumar, “Better algorithms for minimizing average flow-time on related machines,” in International Conference on Automata (ICAT), pp. 181–190, 2006

[4] T. Thanavanich, A. Siri, K. Boonlom, A. Chaikaew, and P. Uthayopas, “Energy-aware scheduling of multiple workflows application on distributed systems,” in International Joint Conference on Computer Science & Software Engineering (JCSSE), pp. 1–6, 2016

[5] H. Topcuoglu, S. Hariri, and M. Y. Wu, “Performance-effective and low-complexity task scheduling for heterogeneous computing,” IEEE Transactions on Parallel and Distributed Systems, vol. 13, no. 3, pp. 260–274, 2002.

[6] H. Li, H. Zhu, G. Ren, H. Wang, H. Zhang, and L. Chen, “Energy-aware scheduling of workflow in cloud center with deadline constraint,” in International Conference on Computational Intelligence and Security (CIS), pp. 415–418, 2016.

[7] H. Tan, Z. Han, X. Li, and F. C. M. Lau, “Online job dispatching and scheduling in edge-clouds,” in IEEE International Conference on Computer Communications (INFOCOM), pp. 1–9, 2017

[8] K. M. Tarplee, A. A. Maciejewski, and H. J. Siegel, “Energy-aware profit maximizing scheduling algorithm for heterogeneous computing systems,” in IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 595–603, 2014.

[9] S. Guo, B. Xiao, Y. Yang, and Y. Yang, “Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing,” in IEEE International Conference on Computer Communications (INFOCOM), pp. 1-9, 2016.

[10] T. Thanavanich, “Energy-aware and performance-aware of workflow application with hybrid scheduling algorithm on cloud computing,” in International Computer Science and Engineering Conference (ICSEC), pp. 1–5, 2019.

[11] M. Sharifi, S. Shahrivari, and H. Salimi, “PASTA: a power-aware solution to scheduling of precedence-constrained tasks on heterogeneous computing resources,” Computing, vol. 95, no. 1, pp. 67–88, 2013.

[12] P. Agrawal and S. Rao, “Energy-aware scheduling of distributed systems,” IEEE Transactions on Automation Science & Engineering, vol. 11, no. 4, pp. 1163–1175, 2014.

[13] W. Yuan and K. Nahrstedt, “Energy-efficient CPU scheduling for multimedia applications.” ACM Transactions on Computer Systems, vol. 24, no. 3, pp. 292-331, 2006.

[14] W. Zheng, B. Emmanuel, and C. Wang, “A randomized heuristic for stochastic workflow scheduling on heterogeneous systems,” in International Conference on Advanced Cloud & Big Data (CBD), pp. 88–95, 2015.

[15] K. Psychas and J. Ghaderi, “Randomized algorithms for scheduling multi-resource jobs in the cloud,” IEEE/ACM Transactions on Networking, vol. 26, no. 5, pp. 2202–2215, 2018.

[16] X. Zhu, L. T. Yang, H. Chen, J. Wang, S. Yin, and X. Liu, “Real-time tasks oriented energy-aware scheduling in virtualized clouds,” IEEE Transactions on Cloud Computing, vol. 2, no. 2, pp. 168–180, 2014.

[17] Z. Yao, Y. Wang, J. Ba, J. Zong, S. Feng, and Z. Wu, “Deadline-aware and energy-efficient dynamic flow scheduling in data center network,” in International Conference on Network & Service Management, pp. 1–4, 2018.

[18] J. Meng, H. Tan, C. Xu, W. Cao, and B. Li, “Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing,” in IEEE International Conference on Computer Communications (INFOCOM), pp. 2287–2295, 2019.

[19] S. Im and B. Moseley, “General profit scheduling and the power of migration on heterogeneous machines,” in ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), pp. 165–173, 2016.

[20] C. Zhang, H. Tan, H. Huang, Z. Han, S. H.-C. Jiang, N. Freris, and X.-Y. Li, “Online dispatching and scheduling of jobs with heterogeneous utilities in edge computing,” in International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiCom), pp. 101–110, 2020.