

Title Scheduling in Serverless Computing for Solar Powered IoT Networks: A Two-Phase Approach

Student Name Junfei Zhan

Student No. 2020101900

Major Information and Computing Science

Supervisor Tengjiao He

Date 20/04/2024

暨南大学

本科生毕业论文

论文题目 太阳能物联网网络无服务器计算中的调度:两阶段方法

学 院 暨南大学伯明翰大学联合学院

学 系 _____

专 业 信息与计算学

姓 名 占俊飞

学 号 2020101900

指导教师 何伟

2024 年 4 月 20 日

Statement of Originality

I hereby declare that the thesis presented is the result of research performed by me personally, under guidance from my supervisor. This thesis does not contain any content (other than those cited with references) that has been previously published or written by others, nor does it contain any material previously presented to other educational institutions for degree or certificate purpose to the best of my knowledge. I promise that all facts presented in this thesis are true and creditable.

Signed: 张俊飞

Date: 2024/04/20

诚信声明

我声明，所呈交的毕业论文是本人在老师指导下进行的研究工作及取得的研究成果。据我查证，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得其他教育机构的学位或证书而使用过的材料。我承诺，论文中的所有内容均真实、可信。

毕业论文作者签名：占俊飞

签名日期：2024年4月20日

Scheduling in Serverless Computing for Solar Powered IoT Networks: A Two-Phase Approach

Abstract: This article explores the integration of serverless and edge computing within solar-powered Internet of Things (IoT) networks. The challenge of optimizing computational task scheduling and resource allocation across IoT devices to enhance efficiency and reduce network strain is addressed. Central to this investigation is the development of a two-phase optimization solution that strategically manages the deployment of computational tasks, taking into account the devices' limited energy, computational power, and storage capacity.

The core contributions of this work include: (i) a novel serverless computing model tailored for energy-constrained IoT environments, ensuring operations within the thresholds of device capabilities while maximizing task efficiency; (ii) a mixed integer linear programming (MILP) formulation that accurately models the dynamics of sub-tasks allocation and resource scheduling, providing a benchmark for optimization; and (iii) a pragmatic two-phase method that approximates the optimal MILP solution, achieving a performance that reaches 81.47% of the MILP's benchmark in terms of application completion rates.

Experimental results highlight the efficacy of the proposed approach in minimizing energy consumption and optimizing task execution, showcasing its potential to significantly improve the operational dynamics of solar-powered IoT networks. The findings not only validate the practical viability of the approach but also illustrate its significant potential to enhance the sustainability, efficiency, and resilience of IoT ecosystems.

Key Words: edge computing, serverless computing, mixed integer linear programming, Two-Phase Approach

太阳能物联网网络无服务器计算中的调度：两阶段方法

摘要：本文探讨了无服务器和边缘计算在太阳能物联网(IoT)网络中的集成。解决了优化物联网设备的计算任务调度和资源分配以提高效率和减少网络压力的挑战。这项研究的核心是开发一种两阶段优化解决方案，该解决方案可以战略性地管理计算任务的部署，同时考虑到设备有限的能量、计算能力和存储容量。

这项工作的核心贡献包括 (i) 为能源受限的物联网环境量身定制的新型无服务器计算模型，可确保在设备能力阈值范围内运行，同时最大限度地提高任务效率；(ii) 混合整数线性规划 (MILP) 公式，可准确模拟子任务分配和资源调度的动态，为优化提供基准；以及 (iii) 一种实用的两阶段方法，该方法近似于最优 MILP 解决方案，在应用完成率方面达到 MILP 基准的 81.47%。

实验结果强调了该方法在最小化能耗和优化任务执行方面的有效性，展示了其显著改善太阳能物联网网络运行动态的潜力。研究结果不仅验证了该方法的实际可行性，还说明了其在提高物联网生态系统的可持续性、效率和弹性方面的巨大潜力。

关键字：边缘计算，无服务器计算，混合整数线性规划，两阶段

Contents

1. Introduction	2
2. Related Works	5
3. System Model	6
3.1 Energy Harvesting Model	7
3.2 Application Model	7
3.3 Communication Model	8
3.4 Device Energy Consumption Model	9
3.5 Problem Formulation	10
4. A Resource-Aware Two-Phase Optimization Solution	11
4.1 Phase One: Function Configuration Decision	11
4.2 Phase Two: sub-task Allocation Decision	13
5. Evaluation	16
5.1 Impact of Number of Devices $ \mathcal{V} $	17
5.2 Impact of Number of subtasks $ \mathcal{S}_i $	18
5.3 Impact of Storage Capacity of Device S_i	20
6. Conclusion	21
Acknowledgements	23
Appendix	24
MILP Solution Code	24
Two-Phase Solution Code	27
Reference	34

1. Introduction

The emergence of smart cities, smart transportation, and precision agriculture has enabled more Internet of Things (IoT) devices to join our lives and are increasingly powered by solar energy to minimize environmental impact. This growth in IoT deployments has accelerated the shift to edge computing, decentralizing data processing to increase efficiency and reduce network stress.

The advent of edge computing has revolutionized data processing by decentralizing the execution of services, accelerating response times, and reducing energy consumption and network congestion. This decentralization plays a crucial role in IoT ecosystems, facilitating rapid data analytics and processing close to user devices, thereby minimizing latency and easing network loads [1],[2]. The integration of edge computing into IoT is further supported by significant industrial investment and the development of standardized Multi-access Edge Computing (MEC) interfaces, highlighting its role in enhancing network services [3].

Alongside the growth of edge computing, serverless computing has emerged as a transformative paradigm, introducing a consumption-based billing model and simplifying server management for developers. This model primarily employed Function as a Service (FaaS) to break down applications into independent, transient, stateless sub-tasks across different nodes [4],[5]. Incorporating serverless architecture into edge ecosystems optimizes resource use and ensures scalability, leading to more dynamic and cost-effective service delivery to end-users [6],[7],[8].

However, merging edge and serverless computing introduces complexity, especially in managing task scheduling and resource allocation within a serverless edge environment [9]. Existing studies provide initial insights but rarely address the detailed scheduling challenges under third-party resource coordination and the limited capacity of edge servers [10],[11]. Our research investigates these scheduling challenges for stateless functions in an IoT edge computing setting, focusing on the detailed decision-making required for energy management, computational limitations, and the availability of functions.

To illustrate this, imagine a smart city scenario where solar-powered IoT devices are deployed on both sides of the road to monitor traffic conditions. When a traffic accident occurs, IoT devices around the accident immediately recognize a new application that needs to be executed - road warning. This road alert application can be broken down into multiple sub-tasks, including real-time traffic data analysis, signal timing of traffic lights and redirect traffic flow,

aiming to quickly respond to accidents and minimize their impact on traffic. Each sub-task is designed to be lightweight and efficient, ensuring rapid execution even on devices with limited computing and energy resources. This process demonstrates how serverless computing can break down a complex application into manageable sub-tasks, each of which needs to be executed by a specific function. This approach not only improves processing efficiency, but also ensures that each sub-task can be processed in a timely and efficient manner, even when resources are limited.

In particular, this example highlights the following key decision points: (i) select which functions to deploy to the IoT devices to perform corresponding sub-tasks based on limited device resources, (ii) schedule which sub-tasks to be processed locally at a specific time, and (iii) schedule which sub-tasks to be uploaded to the cloud server for execution at a specific time. In addition to these considerations, a major optimization goal has emerged: to maximize the number of applications completed by IoT devices in a given time period, ensuring efficient utilization of on-premises and cloud resources. With the flexibility of serverless computing, even resource-constrained IoT devices can effectively participate in the execution of complex applications. This approach allows the device to dynamically choose the best way to perform sub-tasks based on its current capabilities and the urgency of the task, either locally or with computing power in the cloud.

Nevertheless, this system is confronted with several challenges. First, there exists a trade-off between the quantity of functions deployed locally and the storage capacity of IoT devices. More functions deployed locally consume additional storage space, but enable quicker execution of application sub-tasks. Second, the decision-making process is both combinatorial and conditional. Specifically, a local device may either execute sub-tasks locally with corresponding functions deployed or choose to upload tasks to the cloud server. Furthermore, it is necessary to decide the specific time slots for handling and uploading sub-tasks. Assuming N devices, T time slots, and an average of M sub-tasks per application, the solution space is approximately calculated as 2^{TNM^2} , considering each sub-task can be either executed locally or uploaded. Third, devices operate under the uncertainty of energy availability and network conditions, lacking predictive information regarding future energy intake and channel gains to cloud servers[12].

This article explores the scheduling issues within serverless edge computing for IoT, outlining our key contributions as follows:

1. We modeled a new serverless computing system designed for solar-powered IoT net-

works, based on an innovative solar energy harvesting model. This system not only supports charging IoT devices with solar energy but also enables the execution of applications on these devices. Applications are divided into multiple sub-tasks, with each sub-task executed by corresponding stateless functions. Importantly, the execution of these sub-tasks is a collaborative effort between the IoT devices and the cloud server, ensuring adherence to the device's energy, computation, and storage limits.

2. We proposed an optimization problem for serverless computing in IoT environments. This problem aims to maximize the total number of applications successfully executed within a certain period. It considers a set of predetermined function configurations for IoT devices, the execution of sub-tasks locally at specific times, and the uploading of sub-tasks for execution on cloud servers as decision variables, forming a Mixed Integer Linear Programming (MILP) problem.
3. To solve this optimization problem, we designed a two-phase solution. In the first phase, a simplified MILP problem is solved to determine a scheme for the function configuration set of IoT devices. In the second phase, based on the results from the first phase, we use Receding Horizon Control (RHC) and Gaussian Mixture Model (GMM) strategies to decide on the allocation of sub-tasks and the scheme for local or cloud execution at specific times.
4. Our experiments show that this two-phase method significantly improves resource allocation efficiency and application demand satisfaction compared to traditional MILP and random allocation methods. Especially, it dynamically adjusts based on real-time solar energy and channel gain estimates, demonstrating significant performance benefits over random allocation methods as the IoT network scales.

The rest of this article is organized as follows: Section 2 presents a review of the literature, establishing the context and background for our work. Section 3 is dedicated to building the system model and formulating the scheduling problem as a Mixed Integer Linear Programming (MILP) problem. In Section 4, we introduce our two-phase scheduling algorithm, detailing its design and operational principles. Section 5 evaluates the performance of our proposed scheme through extensive simulations, demonstrating its efficacy and advantages. Finally, Section 6 concludes the article, summarizing our findings and suggesting directions for future research.

2. Related Works

Table 1: A COMPARISON OF RELATED WORKS.

Prior works	Energy harvesting	Multiple time slots	Multiple devices	Function configuration
[13],[14]	✓	✗	✗	✓
[9],[15]	✗	✓	✓	✓
[16],[17]	✗	✗	✓	✓
[18],[19]	✗	✗	✗	✓
[20],[21]	✗	✓	✗	✓
This work	✓	✓	✓	✓

Many works have investigated the application of serverless computing within IoT networks, focusing on various aspects such as energy harvesting, time and device management, and function configuration. For instance, Ko *et al.* [13] and Aslanpour *et al.* [14] have delved into energy harvesting and function configuration, revealing potential in optimizing latency and energy consumption in IoT networks. While their contributions are pivotal, they do not extend their frameworks to encompass the complexities of solar energy sources or the dynamics involving multiple devices and time slots.

In addressing the challenges of managing multiple time slots and devices, Xie *et al.* [9] and Deng *et al.* [15] propose multi-objective optimization solutions aimed at minimizing energy consumption, time, and cost. Their research, however, does not integrate energy harvesting techniques, which is a crucial element in promoting sustainability in IoT networks.

The studies by Das *et al.* [16] and Bensalem *et al.* [17] concentrate on optimizing the performance and function allocation for multiple devices without incorporating multiple time slots or energy harvesting, thus limiting the scalability and adaptability of their solutions in dynamic environments.

Furthermore, Cicconetti *et al.* [18] and Bac *et al.* [19] have explored function configuration strategies for serverless computing in edge and IoT networks. Their research, primarily focused on single-device scenarios, underscores the need for advanced function configuration but does not address the challenges posed by energy constraints or the management of multiple devices and time slots.

Vahidinia *et al.* [20] and Agarwal *et al.* [21] have made strides in minimizing latency issues and function cold start frequencies, with an emphasis on time management and efficient function configuration. However, their solutions overlook the integration of energy harvesting and the complexity of handling multiple devices.

These prior works, while foundational, have not fully addressed the unique challenges and opportunities presented by integrating advanced energy harvesting techniques and optimizing function configuration for scalability and efficiency in serverless edge computing frameworks. This work proposes a comprehensive framework that incorporates solar energy harvesting to power multiple devices and optimizes application execution over various time slots with advanced function configuration strategies.

3. System Model

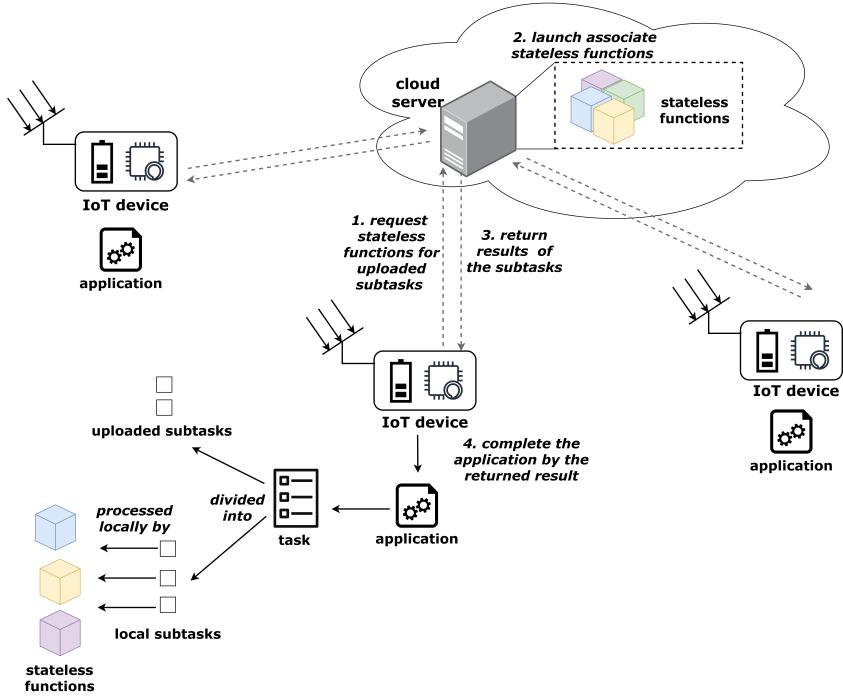


Figure 1: System Model

In this section, we introduce our serverless computing model designed for IoT networks. Our IoT system comprises a cloud server c and a set of solar-powered IoT devices, denoted by \mathcal{V} . Each IoT device $i \in \mathcal{V}$ is equipped with a solar panel and a rechargeable battery. Additionally, every device i hosts an application, which essentially is a task that needs to be computed and is composed of multiple sub-tasks. These sub-tasks are executed by corresponding stateless

functions. The cloud stores a library of functions along with the environment for all functions, represented by \mathcal{F} as the set of functions. Given the limited energy, computation, and storage capacities of the devices, each device can only configure an environment for a subset of functions, denoted by \mathcal{F}_i . To complete an application, device i must request the cloud server s to execute the remaining sub-tasks of the application, since the device can only execute sub-tasks associated with functions within \mathcal{F}_i . We define \mathcal{T} as the set of discrete time slots or the planning horizon. Each time slot is indexed by $t \in \mathcal{T}$ and has a duration of τ .

3.1 Energy Harvesting Model

IoT devices rely solely on solar energy, necessitating efficient management of energy arrivals and storage. Let E_i^t represent the solar energy arriving at device i in time slot t . The device's battery capacity, denoted as B_i , might not accommodate the entire E_i^t , leading to a situation where only a portion, ω_i^t , can be stored during time slot t . The energy available in device i at the beginning of time slot t is given by b_i^t . These variables are subject to the constraints:

$$\omega_i^t \leq E_i^t, \quad \forall i \in \mathcal{V}, \forall t \in \mathcal{T}, \quad (1)$$

$$b_i^t + \omega_i^t \leq B_i, \quad \forall i \in \mathcal{V}, \forall t \in \mathcal{T}. \quad (2)$$

To accurately model the variability of solar energy arrival, we apply a stochastic model. The model simulates E_i^t through a hidden Markov model with states indicating weather conditions: "Excellent", "Good", "Fair", and "Poor", denoted by $w \in \{w_E, w_G, w_F, w_P\}$. Each state has a Gaussian distribution of energy arrivals with mean μ_w and variance σ_w^2 . State transitions are described by a probability matrix \mathbf{P} , where $P_{i,j}$ shows the transition probability from state i to state j . Parameters for these distributions and transitions are based on historical solar irradiance data, as detailed in [22].

3.2 Application Model

In our IoT system, each device $i \in \mathcal{V}$ is dedicated to a single application. The set of sub-tasks for the application on device i is denoted as \mathcal{S}_i . For each sub-task $s \in \mathcal{S}_i$, its computational requirement is determined by its data size D_s (in bits) and its workload W_s (in CPU cycles per bit), leading to a product of $D_s W_s$ which denotes the required CPU cycles for execution.

Each device i has a pre-configured environment for a subset of functions $\mathcal{F}_i \subseteq \mathcal{F}$, where \mathcal{F} is the set of all functions available in the cloud. A binary decision variable $f_{i,k}$ indicates whether

function k is pre-configured on device i , with $f_{i,k} = 1$ if yes, and 0 otherwise. sub-tasks require specific functions for execution. We express this through a binary parameter $\rho_{s,k} = 1$ if sub-task s requires function k , and 0 otherwise. A binary variable $x_{i,s}^t$ denotes whether sub-task s is executed on device i at time slot t . This leads to the following constraint:

$$x_{i,s}^t \leq \sum_{k \in \mathcal{F}_i} \rho_{s,k} f_{i,k}, \quad \forall s \in \mathcal{S}_i, \forall i \in \mathcal{V}, \forall t \in \mathcal{T}, \quad (3)$$

For sub-tasks that cannot be executed locally due to the absence of necessary functions, we introduce a binary variable $y_{i,s}^t$, where $y_{i,s}^t = 1$ if sub-task s is offloaded to the cloud at time t . We establish the relationship between $x_{i,s}^t$ and $y_{i,s}^t$ as:

$$x_{i,s}^t + y_{i,s}^t \leq 1, \quad \forall s \in \mathcal{S}_i, \forall i \in \mathcal{V}, \forall t \in \mathcal{T}, \quad (4)$$

ensuring that each sub-task is either executed locally or offloaded to the cloud, but not both.

The computational capability of device i is denoted by C_i , which limits the number of sub-tasks that can be executed locally at each time slot:

$$\sum_{s \in \mathcal{S}_i} D_s W_s x_{i,s}^t \leq C_i, \quad \forall i \in \mathcal{V}, \forall t \in \mathcal{T}. \quad (5)$$

Additionally, due to limited storage resources, each device can only configure a subset of the functions \mathcal{F}_i . The storage constraint for the stateless functions is given by:

$$\sum_{k \in \mathcal{F}} \sigma_k \cdot f_{i,k} \leq S_i, \quad \forall i \in \mathcal{V}. \quad (6)$$

where S_i denotes the storage capacity of device i , and σ_k represents the storage requirement of function k .

3.3 Communication Model

IoT devices utilize a wireless communication framework for data transmission to the cloud server, where efficiency is governed by channel power gain and transmission energy requirements. The channel power gain between an IoT device i and the cloud server c during a time slot t is given by:

$$g_{i,c}^t = h C_0 \left(\frac{D_{i,c}}{D_0} \right)^{-\alpha}, \quad \forall i \in \mathcal{V}, \forall t \in \mathcal{T} \quad (7)$$

where h represents the channel fading effect, modeled as an exponentially distributed random variable with an average value of one, C_0 is the path loss at a reference distance D_0 , α is the path loss exponent, and $D_{i,c}$ is the Euclidean distance between device i and the cloud server c .

The energy required for an IoT device i to transmit its data to the cloud server in time slot t , denoted as $\lambda_{t,i}^U$, is calculated as:

$$\lambda_{t,i}^U = \sum_{s \in \mathcal{S}_a} \frac{D_s y_{i,s}^t \eta}{g_{i,c}^t}, \quad \forall i \in \mathcal{V}, \forall t \in \mathcal{T} \quad (8)$$

where D_s is the data size of sub-task s in bits, $y_{i,s}^t$ is a binary variable indicating whether the sub-task s is transmitted from device i to the cloud at time t , and η is a constant that represents the energy efficiency of the transmitter. Further, a device can only upload its sub-tasks in slot t when it has sufficient energy for that slot. The energy constraint for a device in slot t is given by:

$$\lambda_{t,i}^U \leq b_t^i, \quad \forall i \in \mathcal{V}, \forall t \in \mathcal{T}, \quad (9)$$

where $\lambda_{t,i}^U$ is the energy required for transmitting sub-tasks, and b_t^i is the available battery energy at the beginning of slot t .

In this section, the wireless communication framework required for IoT devices to transmit data to the cloud server is elaborated upon, taking into account channel power gain and the energy requirements for transmission. Notably, the model does not account for the communication process from the cloud server back to the IoT devices. This decision is based on the consideration that the data size of the results returned to the IoT devices, following the execution of sub-tasks, is typically minimal. Consequently, the impact of this returning data on the overall energy consumption and communication efficiency is negligible.

3.4 Device Energy Consumption Model

Device energy consumption involves two main operations: (i) sub-task upload and (ii) sub-task execution. The energy consumption for sub-task upload has been discussed in Subsection 3.3. For sub-task execution, we model the energy consumption using the data size D_s and workload W_s of the sub-task, along with the energy cost per CPU cycle denoted by ν .

The energy consumed for executing sub-tasks locally on device i is expressed as:

$$\lambda_{t,i}^E = \sum_{s \in \mathcal{S}_a} x_{i,s}^t D_s W_s \nu, \quad \forall i \in \mathcal{V}, \forall t \in \mathcal{T} \quad (10)$$

where $\lambda_{t,i}^E$ represents the energy required to execute local sub-task s on device i . This energy consumption is forced to zero when $x_{i,s}^t = 0, \forall s \in \mathcal{S}_a$. After defining the energy consumption of the two operations on device i , the total energy consumption for device i at time slot t is given

by:

$$\lambda_{t,i}^T = \lambda_{t,i}^E + \lambda_{t,i}^U \quad (11)$$

Furthermore, considering the total energy consumption for both local sub-task execution and sub-task uploading, the cumulative energy consumed for these operations must not exceed the energy harvested by device i over time planning horizon. Therefore, we impose the constraint:

$$\lambda_{t,i}^T \leq \sum_{t \in \mathcal{T}} \omega_t^i, \quad \forall i \in \mathcal{V}, \forall t \in \mathcal{T} \quad (12)$$

where ω_t^i is the energy stored during time slot t .

As a result, the energy level of device i evolves according to the equation:

$$b_t^i = b_{t-1}^i + \omega_{t-1}^i - \lambda_{t,i}^T, \quad \forall t \in \mathcal{T} \quad (13)$$

3.5 Problem Formulation

The primary performance metric of interest is the total number of applications completed within this system over $|\mathcal{T}|$ time slots. The decision-making process involves three main variables for each device i : (i) the function configuration, (ii) the decision to execute sub-tasks locally, and (iii) the decision to upload sub-tasks for cloud offloading. This metric, denoted as Z , quantifies the overall number of applications that are successfully completed throughout the planning horizon \mathcal{T} . Calculation of Z involves aggregating the completion status of all applications, factoring in the execution of all relevant sub-tasks $s \in \mathcal{S}_i$ through either local processing or cloud-based methods.

Within this system, z_i serves as a binary indicator variable for each device i , signifying whether the application on said device is fully executed ($z_i = 1$) or not ($z_i = 0$) over the period \mathcal{T} . To ensure comprehensive application completion, incorporating all its sub-tasks across every time slot, whether executed locally or offloaded to the cloud, the model introduces the following constraint:

$$\sum_{t \in \mathcal{T}} \sum_{s \in \mathcal{S}_i} (x_{i,s}^t + y_{i,s}^t) \geq |\mathcal{T}| |\mathcal{S}_i| z_i, \quad \forall i \in \mathcal{V} \quad (14)$$

This constraint is pivotal for indirectly facilitating the maximization of Z , by setting a prerequisite for the completion status of applications based on the successful execution or offloading of all associated sub-tasks.

The problem can be modeled as the following MILP :

$$\begin{aligned}
 & \underset{\mathbf{f}, \mathbf{x}, \mathbf{y}}{\text{maximize}} \quad Z = \sum_{i \in \mathcal{V}} \sum_{t \in \mathcal{T}} \left(\prod_{s \in \mathcal{S}} (x_{i,s}^t + y_{i,s}^t) \right) = \sum_{i \in \mathcal{V}} z_i \\
 & \text{subject to} \quad (1) - (6), (8) - (14)
 \end{aligned} \tag{15}$$

4. A Resource-Aware Two-Phase Optimization Solution

This section presents a resource-aware two-phase optimization solution addressing problem (15) in this IoT networks. At its core, the solution employs a straightforward strategy that classifies decision variables according to their time-dependence. These variables fall into two primary categories: static variables, which focus on the configuration of functions on devices, and dynamic variables, concerned with the execution of sub-tasks and their distribution between local processing and cloud offloading. Algorithm 1 shows the overall architecture of this two-phase optimization method.

In the first phase, the solution configures the functional capabilities of the IoT devices, ensuring that each is tailored to effectively handle the expected workload within its specific resource constraints. This configuration is static, meaning it remains unchanged over the planning horizon, enabling devices to operate within their energy, storage, and computational limits.

The second phase of the solution capitalizes on the established configurations, intelligently directing the execution of sub-tasks based on real-time evaluations of solar energy availability and channel gain fluctuations. This phase is dynamic, adeptly adjusting to the current state of network resources to optimize sub-task distribution. The objective is to enhance the overall system's productivity by maximizing the number of applications completed, all while adapting to the ever-changing energy and network conditions.

4.1 Phase One: Function Configuration Decision

In the first phase of the solution, a simplified MILP model is deployed to determine the optimal configuration of functions across Internet of Things (IoT) devices. The objective of this phase is to select a subset of functions, \mathcal{F}_i , for each device i from the universal set of functions \mathcal{F} , aiming to strike a balance between the expected workload of each device and its resource constraints, including storage, computational, and energy capacities. The objective of the MILP model is to maximize the coverage of sub-tasks by the configured functions on devices, aiming for the most efficient use of the devices' storage, computational and energy resources.

Algorithm 1 Two-Phase Optimization for IoT Serverless Computing

```

1: procedure Two_Phase_Optimization( $\mathcal{V}, \mathcal{F}, \mathcal{T}$ )
2:   Phase One: Function Configuration
3:   Estimate the average energy arrival  $\bar{E}_i$  for each device  $i$ 
4:   Determine optimal function configuration  $\mathcal{F}_i$  for each device  $i$ 
5:   Phase Two: sub-task Allocation
6:   for each time slot  $t$  in  $\mathcal{T}$  do
7:     Estimate energy arrival  $\hat{E}_i^t$  and channel gain  $\hat{g}_i^t$  for each device  $i$  in time slot  $t$ 
8:     Maximize number of completed applications based on  $\mathcal{F}_i$ 
9:     Update estimation model and energy level  $b_i^t$  for each device  $i$ 
10:  end for
11: end procedure

```

The simplified MILP formulation is as follows:

Objective:

$$\max_{f_{i,k}} \sum_{i \in \mathcal{V}} \sum_{k \in \mathcal{F}} \sum_{s \in \mathcal{S}} \rho_{s,k} f_{i,k} \quad (16)$$

Subject to:

Storage-Aware Constraint:

$$\sum_{k \in \mathcal{F}} \sigma_k f_{i,k} \leq S_i, \quad \forall i \in \mathcal{V} \quad (17)$$

Computation-Aware Constraint:

$$\sum_{k \in \mathcal{F}} \sum_{s \in \mathcal{S}_i} D_s W_s \rho_{s,k} f_{i,k} \leq C_i, \quad \forall i \in \mathcal{V} \quad (18)$$

Energy-Aware Constraint:

$$\sum_{k \in \mathcal{F}} \sum_{s \in \mathcal{S}_i} D_s W_s \nu \rho_{s,k} f_{i,k} \leq \bar{E}_i, \quad \forall i \in \mathcal{V} \quad (19)$$

where:

- $f_{i,k}$ is a binary variable indicating whether function k is configured on device i .
- $\rho_{s,k}$ indicates whether sub-task s requires function k .
- ν denotes the energy cost per CPU cycle

- σ_k represents the storage requirement for function k .
- S_i , C_i , and \bar{E}_i represent the storage, computational, and average energy capacities of device i , respectively.

This formulation ensures that the selected function configurations for each device are within their resource limits, optimizing the network's overall capability to handle its tasks efficiently.

A distinction in this model from the constraints on $f_{i,k}$ in the MILP model, noted in problem (15), is the inclusion of an energy-aware constraint. This constraint accounts for the energy requirements of locally executing sub-tasks with the designated function on each device. It specifically ensures that the energy needed for the function—computed from the energy consumption rate ν per CPU cycle, the data size of the sub-task D_s , and the workload W_s —does not exceed the estimated average energy \bar{E}_i of device i . This method emphasizes the significance of energy efficiency in the sub-task allocation process.

Furthermore, regarding the estimated average energy \bar{E}_i of device i , we utilize a Gaussian Mixture Model (GMM) to estimate the solar energy arrival across all $|\mathcal{T}|$ slots. Then, we calculate the average value of these slots for each device. This approach allows for a more accurate and refined estimation of energy availability, enhancing the model's capability to allocate sub-tasks efficiently.

4.2 Phase Two: sub-task Allocation Decision

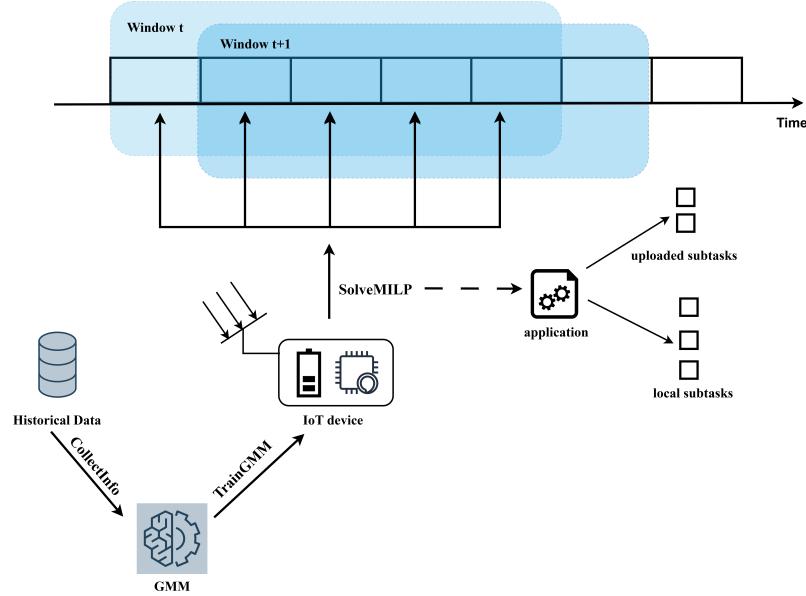


Figure 2: An Overview of Proposed Solution

With the configurations of functions established in Phase One, Phase Two advances the decision-making process for the execution of sub-tasks by leveraging the Receding Horizon Control (RHC) strategy. This method utilizes up-to-date forecasts of solar energy availability and channel gain, enhancing the effectiveness of decisions. Illustrated in Figure 2, a device applies the RHC method to identify the optimal course of action across a decision window of $K = 5$ time slots. Initially, the device executes the action determined for the current time slot t , and then shifts the decision window forward by one slot. Following this adjustment, the system revisits problem (15) to ascertain the optimal action for the next time slot $t + 1$, now incorporating refreshed estimates of channel conditions and solar energy arrivals. This process of continuous updating and reassessment ensures that the decision on whether to process sub-tasks locally or offload them to the cloud is made with the aim of optimizing resource allocation and task execution efficiency.

Our solution involves three stages: (i) Training, (ii) Prediction, and (iii) Decision. Let T be a large positive number. In the Training stage, each device collects historical data on solar energy arrival and channel gain over T slots. These two types of collected data are recorded in sets G_i^T for channel gain and E_i^T for energy harvesting, respectively. This data is then used to train Gaussian Mixture Models (GMMs) for predicting future solar energy arrivals and channel gains. For each IoT device, the historical data collected over T slots are used as input to the GMMs. The training of GMMs is performed via the Expectation-Maximization (EM) algorithm, which iteratively maximizes the likelihood function of the data. During the Expectation step, the algorithm estimates the probabilities that the data points were generated by each component of the mixture model [23]. In the Maximization step, the model parameters are updated to maximize the expected likelihood, refining the model's fit to the data. This iterative process continues until convergence, resulting in a model that can accurately predict future states based on past observations. The Prediction stage involves estimating the future state of these parameters using the trained GMMs, while the Decision stage involves solving the MILP problem of problem (15) without decisions of f_i^t to determine the optimal allocation of sub-tasks.

The solar energy arrival and channel gain are estimated as follows:

$$\hat{E}_i^t = \text{Predict}(\mathcal{E}_i^T), \quad \forall i \in \mathcal{V} \quad (20)$$

$$\hat{g}_i^t = \text{Predict}(\mathcal{G}_i^T), \quad \forall i \in \mathcal{V} \quad (21)$$

In the Algorithm 2, the Training Stage (Lines 4-7) begins with each IoT device i using `CollectInfo` to collect historical solar energy data G_i^T and channel gain data E_i^T . This data is then used to train two GMMs, GMM_{solar} and $GMM_{channel}$, with the `TrainGMM` function. The Prediction Stage (Lines 9-12) involves estimating the solar energy arrival \hat{E}_i^t and channel gain \hat{g}_i^t for the next time slot t for every device i using the trained GMMs. The Decision Stage (Lines 13-18) sets up a decision window \mathcal{K} and employs `SolveMILP` to determine the efficient allocation of sub-tasks within that window. Once the sub-task decisions from Φ for the current time slot t are executed, the energy levels of the devices are updated for the next time slot.

Algorithm 2 sub-task Allocation Using RHC and GMM

```

1: procedure sub-taskAllocation( $\mathcal{V}, \mathcal{T}, K$ )
2:   Initialize: Configure functions  $\mathcal{F}_i$  for each device  $i$ .
3:   — Training Stage —
4:   for  $i \in \mathcal{V}$  do
5:      $\mathcal{G}_i^T, \mathcal{E}_i^T \leftarrow \text{CollectInfo}(i, T)$ 
6:      $GMM_{solar}, GMM_{channel} \leftarrow \text{TrainGMM}(\mathcal{G}_i^T, \mathcal{E}_i^T)$ 
7:   end for
8:   — Prediction Stage —
9:   for  $t \in \mathcal{T}$  do
10:    for  $i \in \mathcal{V}$  do
11:       $\hat{E}_i^t, \hat{g}_i^t \leftarrow \text{Predict}(\mathcal{G}_i^T, \mathcal{E}_i^T, t, K)$ 
12:    end for
13:    — Decision Stage —
14:     $\mathcal{K} \leftarrow \{t, t + 1, \dots, t + K - 1\}$ 
15:     $\Phi \leftarrow \text{SolveMILP}(\mathcal{K}, \{\hat{g}_i^t\}, \{\hat{E}_i^t\})$ 
16:    Execute decisions from  $\Phi$  for time slot  $t$ 
17:    Update the device energy levels  $b_i^{t+1} \leftarrow b_i^t - \text{EnergyUsed}(\Phi)$ 
18:  end for
19: end procedure

```

5. Evaluation

We conducted our evaluation in a Python 3.9 environment, employing Gurobi 9.1.1 to solve the MILP problem delineated in (15). The simulation encompasses a network of IoT devices within a controlled experimental framework. The parameters defining our simulation are:

- Number of time slots (Num_T): 10
- Number of IoT devices (Num_V): 10
- Number of available functions (Num_F): 5
- Number of sub-tasks (Num_Sub): 5
- Battery capacity for each device: 100 Joules (V_Energy) [24]
- Solar panels per device: $30 \times 30 \text{ cm}^2$ at 20% efficiency (V_Solar_Panel) [12]
- Computational capacity per device: 1.5×10^7 CPU cycles per time slot (C_i) [25]
- Storage capacity per device: 700 bits (S_i)
- Data size of sub-tasks: Randomly between 800 to 1500 bits (D_s)
- Workload of sub-tasks: Randomly between 1500 to 3000 CPU cycles per bit (W_s) [9]
- Storage needs for functions: Randomly between 100 to 200 bits (σ_k)
- sub-task function requirements: Binary, randomly determined ($\rho_{s,k}$)
- Transmitter energy efficiency: 0.3 (η)
- Energy per CPU cycle: 2×10^{-6} Joules (ν)
- Storage and battery capacities are constants for simplicity.

Channel gain and solar energy parameters are estimated according to the setup, with devices calibrated to optimize energy and computational efficiency within their operational constraints.

This setup evaluates the performance of our "Two-phase" optimization method relative to a RANDOM approach, which makes sub-task allocations and function configurations decisions arbitrarily. Additionally, the MILP method serves as the benchmark, representing the optimal solution computed by Gurobi 9.1.1. Results presented are the mean of 10 iterations.

5.1 Impact of Number of Devices $|\mathcal{V}|$

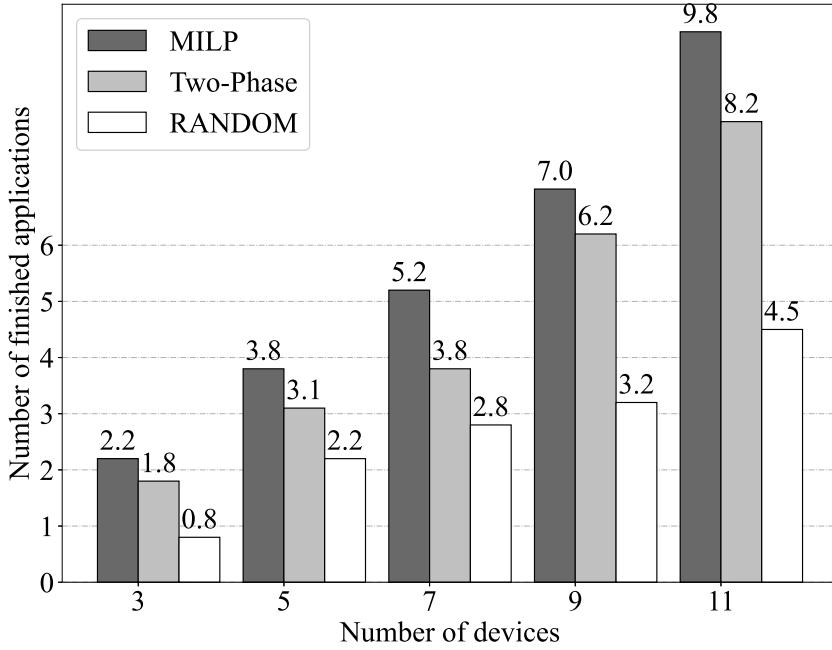


Figure 3: Impact of Number of Devices $|\mathcal{V}|$ on performance

Figure 3 delineates the performance impact based on the number of devices in the system. It is observed that the MILP method consistently exhibits superior performance across all device quantities. The Two-Phase approach follows closely, demonstrating robustness, particularly as the device count increases. In contrast, the RANDOM method shows a noticeable decline in performance with more devices. Notably, with a growing number of devices, the gap in performance between the RANDOM method and the other two approaches becomes increasingly pronounced, underscoring the effectiveness of systematic optimization over random allocation.

The MILP method showcases a consistent increase, with the number of finished applications rising from 2.2 to 9.8 as the number of devices escalates from 3 to 11. This pattern suggests an almost linear scalability, reflective of the method's ability to utilize additional resources effectively. Conversely, the Two-Phase method displays a moderate upswing, with finished applications ascending from 1.8 to 8.2, mirroring a strong but suboptimal scaling compared to MILP. Notably, the gap between MILP and Two-Phase narrows as the device count increases, indicating that the Two-Phase method's performance enhancement has a proportional relationship with system size. On the other hand, the RANDOM method exhibits the most limited progression, with the finished applications count incrementing only marginally from 0.8 to 4.5. This suggests

that while the RANDOM method benefits from additional devices, its lack of strategic planning leads to sublinear and inefficient scaling.

In comparison, the Two-Phase method commences with 81.8% of MILP’s completed applications at three devices, but it experiences a relative improvement as the network size grows. Notably, with 11 devices, the Two-Phase method achieves 83.7% of the MILP’s performance, hinting at its potential to scale robustly within larger networks, despite a lower starting point. This behavior underscores the Two-Phase method’s ability to adapt and improve its relative efficiency as the system expands.

On the contrary, the RANDOM method displays a markedly different trend. Starting at only 36.4% of the MILP’s performance with three devices, it shows a marginal increase in absolute terms but a decreased performance proportionally, with 45.9% at 11 devices. The RANDOM method’s trajectory, marked by a less pronounced upward curve, suggests that it fails to capitalize on the increased network resources efficiently. This inefficiency is evident as its relative gain in performance, compared to MILP, does not correspond proportionally to the increase in the number of devices.

5.2 Impact of Number of subtasks $|\mathcal{S}_i|$

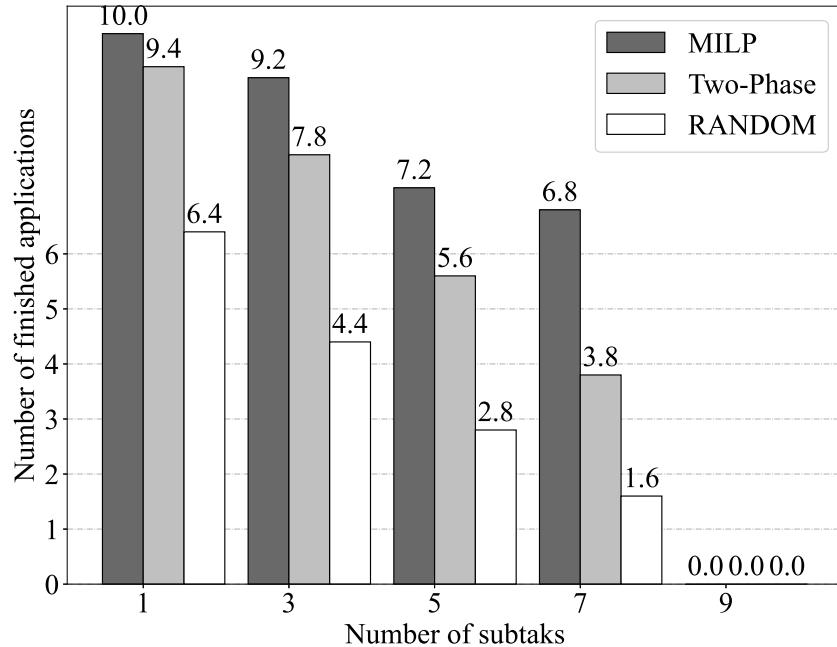


Figure 4: Impact of Number of sub-taks $|\mathcal{S}_i|$ on performance

Figure 4 presents a performance evaluation with varying numbers of sub-tasks $|\mathcal{S}_i|$. It reveals distinctive trends for each of the optimization methods as the number of sub-tasks increases. The MILP method commences with a high performance at 9.4 for a single sub-task, slightly diminishing to 9.2 with three sub-tasks, and then experiencing a more pronounced decrease as the sub-task number rises, reaching a low of 0 at nine sub-tasks. This suggests that while MILP handles smaller sub-task loads with near-perfect efficiency, its performance is inversely proportional to the number of sub-tasks.

The Two-Phase method starts with an excellent performance of 9.4 for a single sub-task, indicating its strength in scenarios with fewer sub-tasks. However, as the sub-task count grows, a decrement in performance is observed, dropping to 7.8 for three sub-tasks and progressively lowering to 0 for nine sub-tasks. This performance trajectory illustrates a resilience to increasing workloads, albeit with a decline in efficacy as the sub-task number becomes substantial. The RANDOM method demonstrates the least effective performance management across the board. It begins with a score of 6.4 and diminishes consistently with each increase in sub-tasks, concluding at 0 for nine sub-tasks. The absence of a strategic approach is evident as the RANDOM method shows the steepest decline in performance, indicating its vulnerability to increased complexity.

The comparison of completion rates presents the Two-Phase method at 94% of MILP's completion rate for a single sub-task, suggesting close competitiveness. Nevertheless, this percentage decreases as the number of sub-tasks grows, with the Two-Phase achieving 84.5%, 77.8%, and 55.9% of MILP's completions for three, five, and seven sub-tasks, respectively. This pattern indicates the Two-Phase method's performance is proportionately affected by the rising sub-task number, reflecting a decrement in efficiency at higher loads compared to MILP's more gradual decline. On the other hand, RANDOM's performance proportions significantly fall from 64% at one sub-task to 25% at seven sub-tasks, which is indicative of its comparative inefficiency in resource allocation and task management under increasing demands.

5.3 Impact of Storage Capacity of Device S_i

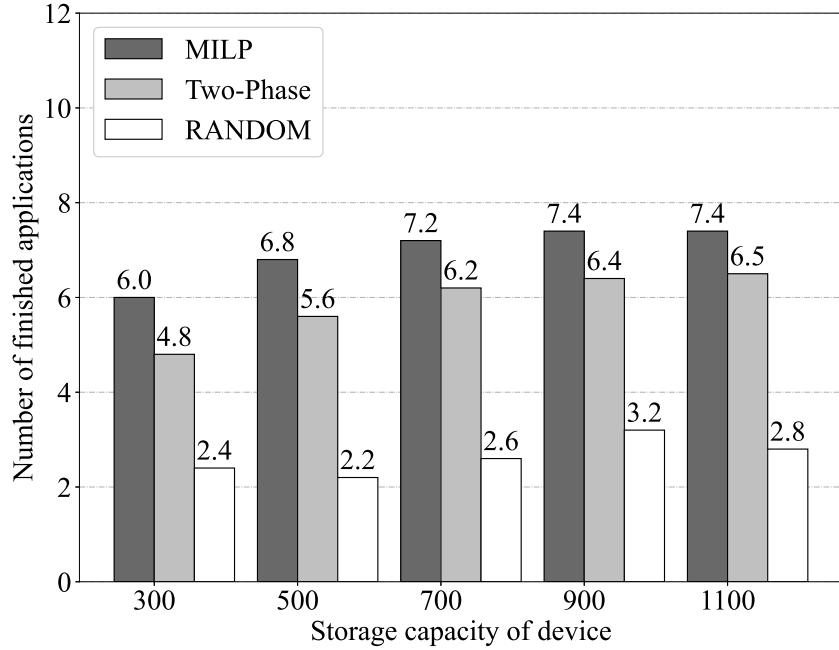


Figure 5: Impact of Storage Capacity of Device S_i on performance

Figure 5 illustrates the correlation between the storage capacity of IoT devices and their performance efficiency. It is evident from the graph that as the storage capacity increases, the number of completed applications by the MILP and Two-Phase methods shows a general upward trend, indicating that larger storage facilitates enhanced performance. The MILP method, in particular, maintains a consistently high number of completed applications across varying storage capacities, suggesting its robustness and efficient utilization of storage resources. In comparison, the Two-Phase method exhibits a gradual increase in performance as storage capacity grows, which implies that while the Two-Phase method benefits from increased storage, it does so with diminishing returns as capacity reaches higher thresholds.

Upon delving into the specifics, the MILP method begins at a completion count of 6 applications for a storage capacity of 300 bits, eventually plateauing at 7.4 applications for capacities of 900 bits and above. This indicates an initial sensitivity to storage which stabilizes at higher capacities. The Two-Phase method also starts at a lower completion count of 4.8 but sees a steadier climb, reaching 6.5 applications at the highest capacity. This suggests a consistent but moderate improvement aligned with storage enhancements. Conversely, the RANDOM method's trajectory is less predictable, with the number of completed applications not following a consistent

pattern as storage capacity increases. Its performance peaks at a capacity of 900 bits with 3.2 completed applications, only to dip slightly at a capacity of 1100 bits.

From the data presented in Figure 5, we observe that the Two-Phase method achieves a commendable proportion of the MILP’s application completion count across various storage capacities. Specifically, at a storage capacity of 300 bits, the Two-Phase method accomplishes 80% of the applications completed by the MILP method. This ratio sees a slight decrease as the storage capacity increases; with the Two-Phase method reaching 88.5% of the MILP’s performance at 500 bits, and gradually narrowing down to 87.8% at a storage capacity of 700 bits. Notably, at the higher capacities of 900 and 1100 bits, the Two-Phase method maintains a consistent completion ratio of 86.5% and 87.8%, respectively, compared to the MILP method.

These ratios reflect the Two-Phase method’s relative efficiency compared to the optimized MILP approach, underscoring its potential as a near-optimal solution. Conversely, when considering the RANDOM method, the completion count reflects more significant fluctuations. At the lowest capacity of 300 bits, RANDOM achieves 40% of the MILP’s completions, while at 900 bits, this performance peaks at 43.2%, demonstrating an inconsistent scaling with storage capacity increases. The RANDOM method, thus, does not exhibit a proportional increase in application completions with enhanced storage, which may be indicative of its inherent inefficiency in resource utilization, particularly at higher storage capacities.

6. Conclusion

This paper has presented a serverless computing model for solar-powered IoT networks that incorporates a novel approach to managing tasks within the constraints of energy, computation, and storage capacities. We introduced a Two-Phase optimization solution, set against a conventional MILP approach and a RANDOM method, to demonstrate the benefits of strategic resource management. Our model, underpinned by a set of IoT devices each equipped with a solar panel and battery, hosts applications that consist of multiple sub-tasks, necessitating efficient execution within the device’s resource constraints.

The MILP approach, representing an optimal solution, and the Two-Phase method, an approximation of the MILP, were both shown to significantly outperform the RANDOM method. As the number of devices increased, the Two-Phase method displayed a proportional relationship in performance relative to MILP, reinforcing its scalability and robustness in larger IoT networks. Averagely, the Two-Phase method attained up to 87.8% of MILP’s performance, a

testament to its near-optimal efficiency and adaptability.

However, the study revealed that the performance of both MILP and Two-Phase approaches deteriorated with an increased number of sub-tasks, eventually declining to zero for nine sub-tasks. This highlights a critical scalability issue as the complexity of task management rises, suggesting a future research direction towards enhancing the Two-Phase method to better manage larger sets of sub-tasks. Moreover, the results indicated a clear dependency on storage capacity, where larger capacities yielded improved performance for MILP and Two-Phase methods, whereas the RANDOM method showed no such correlation, peaking and then diminishing slightly, suggesting inefficient resource utilization.

For future endeavors, our objective is to leverage Directed Acyclic Graphs (DAGs) to model the interdependencies of sub-tasks within the serverless computing framework for IoT networks. By mapping the intricate precedence relations of sub-tasks via DAGs, we anticipate the creation of more sophisticated and scalable optimization strategies. This approach aims to enhance the orchestration of serverless functions, paving the way for more efficient task execution workflows. Embracing DAGs for task modeling is expected to not only improve the granularity of control over the execution order but also to reduce execution latency, thereby advancing the overall efficacy of IoT systems in distributed computing environments.

Acknowledgements

Undergraduate four years along the way, special thanks to all the teachers and classmates encountered in this journey, it is you who shaped my journey unforgettable and glittering, I hope that in the future, I can uphold the kindness and bravery, and continue to go on. In addition, I would like to thank my parents and sister, who have provided me with the most solid protection and support for my life and studies. My family is always the softest and toughest place in my heart. I would also like to express my special thanks to my undergraduate advisor, Prof. Tengjiao He, who not only opened the door of research for me, but also patiently guided me on how to write academic papers, implement algorithms, and how to effectively read the literature. All in all, there is no such thing as an unbroken banquet under the sky, so I hope you all will get better and better, have fun, eat, drink and sleep well in the coming days.

Appendix

This appendix includes the source code developed for the MILP and Two-Phase solutions as part of the above experiments.

MILP Solution Code

The first code snippet represents the Mixed-Integer Linear Programming (MILP) solution approach used for optimizing the given problem. This approach utilizes the Pyomo library for defining and solving the optimization model.

```
1 from pyomo.environ import *
2 import numpy as np
3
4 def call_MILP(V,F,Sub,T,B_i,C_i,S_i,D_s,W_s,sigma_k,rho_s_k,E_i,Dstore_Cap,
5 Ebatt_Cap,eta,g_t,nu):
6
7     # Initialize the model
8     model = ConcreteModel(name="serverless")
9     model.w_i = Var(T, V, bounds=(0, Dstore_Cap))  # $w_{i}^{t}$
10    model.b_i = Var(T, V, bounds=(0, Ebatt_Cap))  # $b_{i}^{t}$
11    # Decision variables
12    model.f = Var(V, F, domain=Binary)  # Function configuration
13    model.x = Var(T, V, Sub, domain=Binary)  # Subtask execution
14    model.y = Var(T, V, Sub, domain=Binary)  # Subtask offloading
15
16    # Constraint (1)
17    def Energy_arrival_device_rule(model, t, i):
18        return model.w_i[t, i] - E_i[t, i] <= 0
19
20    model.Energy_arrival_device = Constraint(T, V, rule=
21 Energy_arrival_device_rule)
22
23    # Constraint (2)
24    def Battery_device_rule(model, t, i):
25        return model.b_i[t, i] + model.w_i[t, i] - B_i[i] <= 0
26
27    model.Battery_device = Constraint(T, V, rule=Battery_device_rule)
```

```

27 # Constraint (3)
28 def application_constraint_rule(model, t, i, s):
29     return model.x[t, i, s] <= sum(rho_s_k[s, k] * model.f[i, k] for k
30     in F)
31
32 model.ApplicationConstraint = Constraint(T, V, Sub, rule=
33     application_constraint_rule)
34
35 # Constraint (4)
36 def subtask_offloading_rule(model, t, i, s):
37     # Ensures a subtask is either executed locally or offloaded, but
38     # not both
39     return model.x[t, i, s] + model.y[t, i, s] <= 1
40
41 model.SubtaskOffloading = Constraint(T, V, Sub, rule=
42     subtask_offloading_rule)
43
44 # Constraint (5)
45 def computational_capacity_rule(model, t, i):
46     # Sum of CPU cycles required by all subtasks must not exceed the
47     # device's computational capacity
48     return sum(D_s[s] * W_s[s] * model.x[t, i, s] for s in Sub) <= C_i[
49         i]
50
51 model.ComputationalCapacity = Constraint(T, V, rule=
52     computational_capacity_rule)
53
54 # Constraint (6)
55 def storage_capacity_rule(model, i):
56     # Sum of storage requirements of all functions configured on device
57     # must not exceed its storage capacity
58     return sum(sigma_k[k] * model.f[i, k] for k in F) <= S_i[i]
59
60 model.StorageCapacity = Constraint(V, rule=storage_capacity_rule)
61
62 # Constraint (9)
63 def energy_consumption_execution_rule(model, t, i):
64     # Energy consumed for executing subtasks locally on device i at
65     # time t

```

```

57     return sum(model.y[t, i, s] * D_s[s] * W_s[s] * eta/g_t[t,i] for s
58     in Sub) <= model.b_i[t, i]
59
60     model.EnergyConsumptionExecution = Constraint(T, V, rule=
61     energy_consumption_execution_rule)
62
63     # Constraint (12)
64     def energy_consumption_total_rule(model, t, i):
65         # Energy consumed for executing subtasks locally on device i at
66         # time t
67
68         return sum(model.y[t, i, s] * D_s[s] * W_s[s] * eta/g_t[t,i] for s
69         in Sub) + sum(model.x[t, i, s] * D_s[s] * W_s[s] * nu for s in Sub) <=
70         sum(model.w_i[t, i] for t in T)
71
72     model.EnergyConsumptiontotal = Constraint(T, V, rule=
73     energy_consumption_total_rule)
74
75     # Constraint (13)
76     def battery_state_update_rule(model, t, i):
77         # This constraint updates the battery state based on previous state
78         # energy harvested, and energy consumed
79
80         # Assuming energy consumed for transmission ( $\lambda^U_{t,i}$ ) is
81         # calculated or defined elsewhere
82
83         if t == 0: # Initial condition
84             return Constraint.Skip # Or define an initial state for model.
85             b_i[0, i]
86         else:
87             return model.b_i[t, i] == model.b_i[t - 1, i] + model.w_i[t -
88             1, i] - (sum(model.y[t, i, s] * D_s[s] * W_s[s] * eta/g_t[t,i] for s in
89             Sub) + sum(model.x[t, i, s] * D_s[s] * W_s[s] * nu for s in Sub))
90
91     model.BatteryStateUpdate = Constraint(T, V, rule=
92     battery_state_update_rule)
93
94     #Object
95     # Decision variables for application completion
96     model.z = Var(V, domain=Binary) # Application completion indicator for
97     each device
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
825
826
827
827
828
829
829
830
831
832
833
833
834
835
835
836
837
837
838
839
839
840
841
841
842
843
843
844
845
845
846
847
847
848
849
849
850
851
851
852
853
853
854
855
855
856
857
857
858
859
859
860
861
861
862
863
863
864
865
865
866
867
867
868
869
869
870
871
871
872
873
873
874
875
875
876
877
877
878
879
879
880
881
881
882
883
883
884
885
885
886
887
887
888
889
889
890
891
891
892
893
893
894
895
895
896
897
897
898
899
899
900
901
901
902
903
903
904
905
905
906
907
907
908
909
909
910
911
911
912
913
913
914
915
915
916
917
917
918
919
919
920
921
921
922
923
923
924
925
925
926
927
927
928
929
929
930
931
931
932
933
933
934
935
935
936
937
937
938
939
939
940
941
941
942
943
943
944
945
945
946
947
947
948
949
949
950
951
951
952
953
953
954
955
955
956
957
957
958
959
959
960
961
961
962
963
963
964
965
965
966
967
967
968
969
969
970
971
971
972
973
973
974
975
975
976
977
977
978
979
979
980
981
981
982
983
983
984
985
985
986
987
987
988
989
989
990
991
991
992
993
993
994
995
995
996
997
997
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1650
1651
1651
1652
1652
1653
1653
1654
1654
1655
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1660

```

```

83     # Constraint (14) to ensure subtask completion leads to application
84     # completion
85     def application_completion_rule(model, i):
86         # For each device i, ensure that if all subtasks are either
87         # executed or offloaded, then z[i] can be 1
88         return sum(model.x[t, i, s] + model.y[t, i, s] for t in T for s in
89         Sub) >= len(T) * len(Sub) * model.z[i]
90
91     model.ApplicationCompletion = Constraint(V, rule=
92         application_completion_rule)
93
94     # Objective: Maximize the number of completed applications
95     def objective_rule(model):
96         return sum(model.z[i] for i in V)
97
98     model.objective = Objective(rule=objective_rule, sense=maximize)
99
100    # Solve the MILP
101    opt = SolverFactory('gurobi', solver_io='python')
102    results = opt.solve(model) # solves and updates instance
103    Max_min_Opt = value(model.objective)
104    model.f.pprint()
105    return Max_min_Opt

```

Two-Phase Solution Code

The second code snippet outlines the Two-Phase solution method. The first phase deals with the function configuration, while the second phase focuses on decision-making based on the function configurations derived from the first phase.

```

1 #Phase 1
2 def call_function_config_MILP(V, F, S_i, C_i, sigma_k, rho_s_k, D_s, W_s,
3     Sub):
4     # Initialize the model
5     model = ConcreteModel(name="FunctionConfiguration")
6
7     # Decision variables
8     model.f = Var(V, F, domain=Binary) # Function configuration

```

```

9  # Constraints
10 # Constraint (1): Storage capacity constraint for each device
11 def storage_capacity_rule(model, i):
12     return sum(sigma_k[k] * model.f[i, k] for k in F) <= S_i[i]
13 model.StorageCapacity = Constraint(V, rule=storage_capacity_rule)
14
15 # Constraint (2) - Consider computational capacity if necessary
16 # the computational demand (C_demand) of having a function on a device.
17 def computational_capacity_rule(model, i):
18     C_demand = {k: W_s[k] * D_s[k] for k in F} # Example way to
19     calculate demand, adjust as needed
20     return sum(C_demand[k] * model.f[i, k] for k in F) <= C_i[i]
21 model.ComputationalCapacity = Constraint(V, rule=
22 computational_capacity_rule)
23
24 # Objective: Maximize the coverage of subtasks by the configured
25 # functions on devices
26 def objective_rule(model):
27     return sum(sum(rho_s_k.get((s, k), 0) * model.f[i, k] for k in F)
28 for s in Sub for i in V)
29 model.objective = Objective(rule=objective_rule, sense=maximize)
30
31 # Solve the MILP
32 solver = SolverFactory('gurobi')
33 results = solver.solve(model, tee=True)
34
35 # Extract and return the function configuration solution
36 f_sol = np.zeros((len(V), len(F)))
37 for i in range(len(V)):
38     for j in range(len(F)):
39         f_sol[i, j] = model.f[i, j].value
40 # f_solution = {(i, k): value(model.f[i, k]) for i in V for k in F}
41 return f_sol
42
43
44
45
46 #Phase 2
47 def MPC_call_MILP(ban_i, energy_level, energy_store, x_dec, y_dec, current_slot,
48 f_known, V, F, Sub, T, B_i, C_i, S_i, D_s, W_s, sigma_k, rho_s_k, E_i, Dstore_Cap,
49 Ebatt_Cap, eta, g_t, nu):

```

```

42
43 # Initialize the model
44 model = ConcreteModel(name="serverless")
45 model.w_i = Var(T, V, bounds=(0, Dstore_Cap)) #  $w_i^t$ 
46 model.b_i = Var(T, V, bounds=(0, Ebatt_Cap)) #  $b_i^t$ 
47 # Decision variables
48 model.x = Var(T, V, Sub, domain=Binary) # Subtask execution
49 model.y = Var(T, V, Sub, domain=Binary) # Subtask offloading
50 # Fixed function configuration from Phase 1
51 f_init = {(v, f): f_known[v, f] for v in range(len(V)) for f in range(len(F))}
52 model.f = Param(V, F, initialize=f_init, within=Binary)
53 if current_slot > 0:
54     # update the record information
55     for v in range(len(V)):
56         model.b_i[0, v].fix(energy_level[v]) # Fixed battery status
57         model.w_i[0, v].fix(energy_store[v]) # Fixed energy store
58         for s in range(len(Sub)):
59             model.x[0, v, s].fix(x_dec[v, s]) # Fixed subtask
60             execution
61             model.y[0, v, s].fix(y_dec[v, s]) # Fixed subtask
62             offloading
63             # Ban the device that finish its device
64             for i in range(len(V)):
65                 if ban_i[i] == 1:
66                     # Fix all decision variables related to the banned device to zero
67                     for t in range(len(T)):
68                         for s in range(len(Sub)):
69                             model.x[t, i, s].fix(0) # Do not execute any tasks
70                             model.y[t, i, s].fix(0) # Do not offload any tasks
71
72             def ban_x_rule(model, t, i, s):
73                 if ban_i[i] == 1:
74                     return model.x[t, i, s] == 0
75                 else:
76                     return Constraint.Skip
77
78             model.ban_x_con = Constraint(T, V, Sub, rule=ban_x_rule)

```

```

78  def ban_y_rule(model, t, i, s):
79      if ban_i[i] == 1:
80          return model.y[t, i, s] == 0
81      else:
82          return Constraint.Skip
83
84  model.ban_y_con = Constraint(T, V, Sub, rule=ban_y_rule)
85
86  # Constraint (1)
87  def Energy_arrival_device_rule(model, t, i):
88      return model.w_i[t, i] - E_i[t, i] <= 0
89
90  model.Energy_arrival_device = Constraint(T, V, rule=
91      Energy_arrival_device_rule)
92
93  # Constraint (2)
94  def Battery_device_rule(model, t, i):
95      return model.b_i[t, i] + model.w_i[t, i] - B_i[i] <= 0
96
97  model.Battery_device = Constraint(T, V, rule=Battery_device_rule)
98
99  # Constraint (3)
100 def application_constraint_rule(model, t, i, s):
101     return model.x[t, i, s] <= sum(rho_s_k[s, k] * model.f[i, k] for k
102     in F)
103
104  model.ApplicationConstraint = Constraint(T, V, Sub, rule=
105      application_constraint_rule)
106
107  # Constraint (4)
108  def subtask_offloading_rule(model, t, i, s):
109      # Ensures a subtask is either executed locally or offloaded, but
110      # not both
111      return model.x[t, i, s] + model.y[t, i, s] <= 1
112
113  model.SubtaskOffloading = Constraint(T, V, Sub, rule=
114      subtask_offloading_rule)
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
826
827
827
828
829
829
830
831
832
833
834
835
836
837
837
838
839
839
840
841
842
843
844
845
846
846
847
848
848
849
849
850
851
852
853
854
855
856
856
857
858
858
859
859
860
861
862
863
864
865
866
866
867
868
868
869
869
870
871
872
873
874
875
876
876
877
878
878
879
879
880
881
882
883
884
885
886
886
887
888
888
889
889
890
891
892
893
894
894
895
896
896
897
897
898
898
899
899
900
901
902
903
904
904
905
906
906
907
907
908
908
909
909
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1650
1651
1651
1652
1652
1653
1653
1654
1654
1655
1655
1656
1656
1
```



```

140 # Constraint (13)
141 def battery_state_update_rule(model, t, i):
142     # This constraint updates the battery state based on previous state
143     # energy harvested, and energy consumed
144     # Assuming energy consumed for transmission ( $\lambda^U_{t,i}$ ) is
145     # calculated or defined elsewhere
146     if t == 0: # Initial condition
147         return Constraint.Skip # Or define an initial state for model.
148         b_i[0, i]
149     else:
150         return model.b_i[t, i] == model.b_i[t - 1, i] + model.w_i[t - 1, i] -
151             (sum(model.y[t, i, s] * D_s[s] * W_s[s] * eta/g_t[t,i] for s in
152                 Sub) + sum(model.x[t, i, s] * D_s[s] * W_s[s] * nu for s in Sub))
153
154 model.BatteryStateUpdate = Constraint(T, V, rule=
155     battery_state_update_rule)
156
157 #Object
158 # Decision variables for application completion
159 model.z = Var(V, domain=Binary) # Application completion indicator for
160 each device
161
162 # Constraint (14) to ensure subtask completion leads to application
163 completion
164 def application_completion_rule(model, i):
165     # For each device i, ensure that if all subtasks are either
166     # executed or offloaded, then z[i] can be 1
167     return sum(model.x[t, i, s] + model.y[t, i, s] for t in T for s in
168                 Sub) >= len(T) * len(Sub) * model.z[i]
169
170 model.ApplicationCompletion = Constraint(V, rule=
171     application_completion_rule)
172
173 # Objective: Maximize the number of completed applications
174 def objective_rule(model):
175     return sum(model.z[i] for i in V)
176
177 model.objective = Objective(rule=objective_rule, sense=maximize)

```

```

168 # Solve the MILP
169 opt = SolverFactory('gurobi', solver_io='python')
170 results = opt.solve(model) # solves and updates instance
171 Max_min_Opt = value(model.objective)
172
173 # return necessary information
174 dec_slot = 1
175 energy_levelc = np.zeros(len(V))
176 energy_storec = np.zeros(len(V))
177 x_decc = np.zeros((len(V), len(Sub)))
178 y_decc = np.zeros((len(V), len(Sub)))
179 com_app = np.zeros(len(V))
180 for i in range(len(V)):
181     energy_levelc[i] = model.b_i[dec_slot, i].value
182     energy_storec[i] = model.w_i[dec_slot, i].value
183     for j in range(len(Sub)):
184         x_decc[i, j] = model.x[dec_slot, i, j].value
185         y_decc[i, j] = model.y[dec_slot, i, j].value
186
187 x_judge = np.zeros((2, len(V), len(Sub)))
188 y_judge = np.zeros((2, len(V), len(Sub)))
189 for ii in range(2):
190     for jj in range(len(V)):
191         for kk in range(len(Sub)):
192             x_judge[ii, jj, kk] = model.x[ii, jj, kk].value
193             y_judge[ii, jj, kk] = model.y[ii, jj, kk].value
194 for ie in range(len(V)):
195     # Calculate the product of the concatenation of (x_i^t + y_i^t) on
196     # subtask s for all time steps for device i
197     product_along_subtasks = np.prod(x_judge[:, ie, :], y_judge[:, ie, :],
198                                     axis=1) # Consecutive multiplication over subtask dimensions
199     # The results are then summed over all time steps
200     product_sum = np.sum(product_along_subtasks)
201     com_app[ie] = product_sum >= 1
202
203 return com_app, energy_levelc, energy_storec, x_decc, y_decc

```

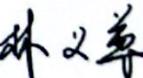
Reference

- [1] M. Campbell. “Smart edge: The center of data gravity out of the cloud”. In: *Computer* 52 (Dec. 2019), pp. 99–102.
- [2] W. Duan, J. Gu, M. Wen, G. Zhang, Y. Ji, and S. Mumtaz. “Emerging technologies for 5G-IoV networks: Applications, trends and opportunities”. In: *IEEE Netw.* 34.5 (Sept. 2020), pp. 283–289.
- [3] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella. “On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration”. In: *IEEE Commun. Surveys Tuts.* 19.3 (2017), pp. 1657–1681.
- [4] P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski. “The rise of serverless computing”. In: *Commun. ACM* 62.12 (Nov. 2019), pp. 44–54.
- [5] S. Hendrickson, S. Sturdevant, T. Harter, V. Venkataramani, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. “Serverless computation with openlambda”. In: *Proc. 8th USENIX Conf. Hot Topics Cloud Comput.* 2016, pp. 33–39.
- [6] C. Cicconetti, M. Conti, A. Passarella, and D. Sabella. “Toward distributed computing environments with serverless solutions in edge systems”. In: *IEEE Commun. Mag.* 58.3 (Mar. 2020), pp. 40–46.
- [7] R. Xie, Q. Tang, S. Qiao, H. Zhu, F. R. Yu, and T. Huang. “When serverless computing meets edge computing: Architecture, challenges, and open issues”. In: *IEEE Wireless Commun.* 28.5 (Oct. 2021), pp. 126–133.
- [8] T. Rausch, A. Rashed, and S. Dustdar. “Optimized container scheduling for data-intensive serverless edge computing”. In: *Future Gener. Comput. Syst.* 114 (2021), pp. 259–271.
- [9] R. Xie, D. Gu, Q. Tang, T. Huang, and F. R. Yu. “Workflow Scheduling in Serverless Edge Computing for the Industrial Internet of Things: A Learning Approach”. In: *IEEE Transactions on Industrial Informatics* 19.7 (July 2023), pp. 8242–8252. doi: 10.1109/TII.2022.3217477.
- [10] C. Cicconetti, M. Conti, and A. Passarella. “A decentralized framework for serverless edge computing in the Internet of Things”. In: *IEEE Trans. Netw. Service Manag.* 18.2 (June 2021), pp. 2166–2180.

- [11] A. Glikson, S. Nastic, and S. Dustdar. “Deviceless edge computing: Extending serverless computing to the edge of the network”. In: *Proc. 10th ACM Int. Syst. Storage Conf. (SYSTOR)*. 2017, p. 28.
- [12] T. He, K.-W. Chin, H. Ren, and X. Liu. “Maximizing Virtual Network Embedding Requests in RF-Charging IoT Networks”. In: *IEEE Commun. Lett.* 26.4 (Apr. 2022), pp. 863–867.
- [13] H. Ko, S. Pack, and V. C. M. Leung. “Performance Optimization of Serverless Computing for Latency-Guaranteed and Energy-Efficient Task Offloading in Energy-Harvesting Industrial IoT”. In: *IEEE Internet Things J.* 10.3 (June 2023), pp. 1897–1907.
- [14] M. S. Aslanpour, A. N. Toosi, M. A. Cheema, and R. Gaire. “Energy-Aware Resource Scheduling for Serverless Edge Computing”. In: *Proc. 22nd IEEE Int. Symp. Cluster Cloud Internet Comput. (CCGrid)*. June 2022, pp. 190–199.
- [15] S. Deng, H. Zhao, Z. Xiang, C. Zhang, R. Jiang, Y. Li, J. Yin, S. Dustdar, and A. Y. Zomaya. “Dependent Function Embedding for Distributed Serverless Edge Computing”. In: *IEEE Trans. Parallel Distrib. Syst.* 33.10 (Oct. 2022), pp. 2346–2357.
- [16] A. Das, S. Imai, S. Patterson, and M. P. Wittie. “Performance Optimization for Edge-Cloud Serverless Platforms via Dynamic Task Placement”. In: *Proc. 20th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*. May 2020, pp. 41–50.
- [17] M. Bensalem, F. Carpio, and A. Jukan. “Towards Optimal Serverless Function Scaling in Edge Computing Network”. In: *Proc. IEEE Int. Conf. Commun. (ICC)*. May 2023, pp. 828–833.
- [18] C. Cicconetti, M. Conti, and A. Passarella. “A Decentralized Framework for Serverless Edge Computing in the Internet of Things”. In: *IEEE Trans. Netw. Serv. Manag.* 18.2 (June 2021), pp. 2166–2180.
- [19] T. P. Bac, M. N. Tran, and Y. Kim. “Serverless Computing Approach for Deploying Machine Learning Applications in Edge Layer”. In: *Proc. Int. Conf. Inf. Netw. (ICOIN)*. Jan. 2022, pp. 396–401.
- [20] P. Vahidinia, B. Farahani, and F. Shams Aliee. “Mitigating Cold Start Problem in Serverless Computing: A Reinforcement Learning Approach”. In: *IEEE Internet Things J.* 10.5 (May 2023), pp. 3917–3927.

- [21] S. Agarwal, M. A. Rodriguez, and R. Buyya. “A Reinforcement Learning Approach to Reduce Serverless Function Cold Start Frequency”. In: *Proc. IEEE/ACM 21st Int. Symp. Cluster Cloud Internet Comput. (CCGrid)*. May 2021, pp. 797–803.
- [22] M.-L. Ku, Y. Chen, and K. J. R. Liu. “Data-Driven Stochastic Models and Policies for Energy Harvesting Sensor Communications”. In: *IEEE J. Sel. Areas Commun.* 33.8 (Aug. 2015), pp. 1505–1520.
- [23] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [24] J. Zhan, J. Wu, T. He, and K.-W. Chin. “Task Offloading and Approximate Computing in Solar Powered IoT Networks”. In: *IEEE Netw. Lett.* 6.1 (Mar. 2024), pp. 26–30.
- [25] J. Tang, W. P. Tay, T. Q. S. Quek, and B. Liang. “System Cost Minimization in Cloud RAN With Limited Fronthaul Capacity”. In: *IEEE Trans. Wireless Commun.* 16.5 (May 2017), pp. 3371–3384.

本科生毕业设计（论文）评定表

指导教师意见	论文评语:	In the thesis, the author considers a serverless computing-based IoT system. He proposes a novel problem of maximize the total number of executed applications, and models the problem as an MILP. After that, the authors designs a Receding Horizon Control-based solution. The experimental results demonstrate the effectiveness of the proposed method. This thesis is written well, and the organization structure is proper. Overall, the thesis meets the requirements for undergraduate dissertation.	
	评定分数:	95	签章: 
评阅人意见	论文评语:	占俊飞同学的毕业论文提出了一种两阶段方法用于太阳能物联网网络无服务器计算中的调度, 通过数值实验证明了所提出方法的有效性。论文结构清晰, 观点明确, 论据充分, 对实际应用具有一定的参考价值。该论文能达到学士学位论文的要求, 同意进行答辩。	
	评定分数:	91	签章: 
学院意见	该学生答辩评分成绩为 92, 按照规定的记分权重计算出的最终成绩为 92, 评分等级为优秀。		
	分数:	92/100	签章: 